Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DEL SOFTWARE

APLICACIÓN WEB PARA LA VISUALIZACIÓN Y CLASIFICACIÓN DE BOTS POLÍTICOS EN TWITTER

Estudiante: Carlos Fraiz Ares
Dirección: Patricia Martín Rodilla

A Coruña, septiembre de 2020.

Resumen

En este Trabajo de Fin de Grado se ha desarrollado una aplicación web para la visualización y clasificación de bots políticos en Twitter.

Este trabajo está motivado por la creciente influencia del software en la opinión pública a través del conjunto de medios sociales, agentes autónomos y algoritmos. La incidencia de estas herramientas software en cualquier proceso consultivo o electoral es de especial interés para analistas de medios de comunicación, politólogos o socíologos (entre otros profesionales), que conforman los usuarios principales de la aplicación. Estos profesionales necesitan de herramientas software y sistemas de visualización que les permitan realizar análisis de grandes conjuntos de información de manera ágil y adaptada a sus preguntas de investigación y análisis. Con este fin, en el marco de una colaboración interuniversitaria, se desarrolló un clasificador que por medio de procesamiento del lenguaje natural determina la probabilidad de que una cuenta de la red social Twitter sea un programa informático (bot) o un usuario real.

La aplicación web a desarrollar adapta el clasificador a entornos web modernos, haciéndolo accesible a los usuarios finales y desarrollando sistemas de visualización de información (en adelante, *InfoVis*) adaptados a la informacion que proporciona dicho clasificador. La InfoVis es una disciplina que consiste en el uso de representaciones visuales de datos para reforzar la capacidad cognitiva del usuario [1], [2].

Para el desarrollo del proyecto se ha seguido una adaptación de la metodología de desarrollo ágil Scrum, se ha implementado utilizando tecnologías de código abierto como MySQL, Spring, Hibernate, Maven, React y se han empleado herramientas de código abierto como Eclipse, Visual Studio Code, GitLab o Taiga para facilitar el desarrollo.

Palabras clave:

- React
- Redux
- Aplicación web
- Java
- Spring
- Hibernate
- MySql
- Bootstrap
- REST
- Maven
- GitLab
- Taiga
- InfoVis
- d3.js
- Tweet
- Retweet

Índice general

Índice de figuras

Capítulo 1
Introducción

En este capítulo se definen los objetivos y la motivación detrás de este proyecto, así como sus líneas maestras.

1.1 Motivación

La manipulación de la opinión pública a través de las redes sociales se ha convertido en un problema real para la política del siglo XXI. Alrededor del mundo, partidos políticos, agencias gubernamentales y otros agentes políticos han usado este medio para difundir noticias falsas y desinformación (*"fake news"* [4]), ejerciendo control y censura y menoscabando la credibilidad de los medios de comunicación, la ciencia y las instituciones públicas [5], [6].

La tecnología moderna permite rastrear datos personales y trazar con ellos perfiles ideológicos de los ciudadanos, pudiendo construir y enviar mensajes personalizados diseñados para influir en cada uno de los perfiles obtenidos.

Estudios recientes de la universidad de Oxford [7] muestran evidencias de campañas de manipulación en 48 países en los que por lo menos un agente político usaba redes sociales como medio para la manipulación de la opinión pública. [8]

Podemos observar el gran número de agentes políticos que intervienen en la manipulación de las redes sociales (Figura 1.1), ya sea usando programas informáticos (cuentas automatizadas o bots), personas usando un conjunto de cuentas falsas o bien cuentas con ciertos procesos automatizados pero con algún tipo de intervención humana para simular cuentas legítimas.

En el contexto de este proyecto se pretende adaptar a entornos web y facilitar el uso de un clasificador de cuentas de la red social Twitter para usuarios con perfil no tecnológico

Country	Fake Account Type	Pro-Government or Party Messages	Attacks on the Opposition	Distracting or Neutral Messages	Trolling or Harassment
Angola			■		
Argentina		■	■		■
Armenia				■	
Australia		■	■		
Austria			■		■
Azerbaijan		■	■	■	■
Bahrain			■		■
Brazil		■	■	■	■
Cambodia					
China		■	■	■	■
Colombia			■		■
Cuba			■	■	■
Czech Republic				■	
Ecuador		■	■		■
Egypt			■		■
Germany		■	■		■
Hungary		■	■		■
India		■	■		■
Iran		■	■	■	
Israel		■			
Italy		■	■		
Kenya			■		
Kyrgyzstan		■	■		
Malaysia		■	■		
Mexico		■	■		■
Myanmar		■	■		
Netherlands					
Nigeria			■		
North Korea		■	■		
Pakistan		■	■		
Philippines		■	■		■
Poland			■		■
Russia		■	■	■	■
Saudi Arabia		■			
Serbia		■	■		■
South Africa		■	■		■
South Korea		■	■		■
Syria		■	■		■
Taiwan		■	■	■	■
Thailand					
Turkey		■	■		■
Ukraine		■	■		■
UAE		■	■		■
United Kingdom		■	■	■	■
United States		■	■	■	■
Venezuela		■	■	■	
Vietnam		■	■		
Zimbabwe			■		

Source: Authors' evaluations based on data collected. **Note:** This table reports on the messaging and valence strategies of cyber troops. A filled box indicates evidence found. For fake account types: = human accounts; = automated accounts = cyborg accounts; = no evidence found.

Figura 1.1: Estrategias usadas para la manipulación en redes sociales

(analistas de médios de comunicación y politólogos, entre otros). Debido a la gran cantidad de información manejada en el análisis de redes sociales, tanto a nivel de datos de entrada (número de cuentas y relaciones entre ellas), como a nivel de salida (probabilidades de clasificación bot/humano), así como los perfiles de usuarios detectados, es necesario diseñar y desarrollar visualizaciones que permitan a los usuarios analizar estas grandes cantidades de datos de manera conjunta y persistiendolos para su reusabilidad.

La aplicación a desarrollar hace especial hincapié en la visualización de la información para aportar valor a los expertos y facilitar su trabajo de análisis y exploración de los datos.

1.1.1 Estado del arte

La visualización de información es el uso de representaciones visuales de datos para reforzar la capacidad cognitiva del usuario [9]. Es un campo de conocimiento que ha ganado interés y relevancia gracias a campos como el Big Data.

En el campo de la influencia política a través de las redes sociales existen todo tipo de investigaciones y documentos que contienen representaciones de grandes conjunto de datos. En la figura 1.2 se puede ver un ejemplo de [3] analizando perfiles de cuentas de Twitter durante las elecciones catalanas. [10].

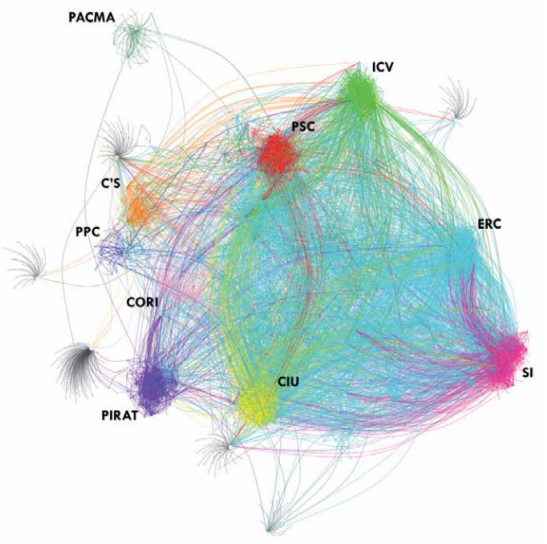


Figura 1.2: Visualización de grafos para retweets entre cuentas con perfiles ideológicos definidos durante el periodo de las elecciones catalanas. Imagen de [3]

Teniendo en cuenta las técnicas de visualización de la información, la aplicación web a desarrollar usará la información relativa a las cuentas para mostrar distintos datos para los expertos, tratando de responder a sus preguntas de investigación e incorporando los resultados del clasificador de manera interactiva.

Existen numerosos proyectos que a través del procesamiento de lenguaje natural [11] (en adelante *nlp*) obtienen algoritmos de clasificación para la detección de bots tóxicos en el dominio político desde Twitter. Los trabajos en este ámbito con resultado satisfactorio se han realizado solo para textos de entrada en inglés.

El clasificador del proyecto de investigación que engloba este TFG usa técnicas de *hybrid intelligence* [12] [13] para *nlp* y es el primero que lo hace para entradas en español. *hybrid intelligence* consiste es una arquitecturas que integra información simbólica en modelos estadísticos o neuronales para permitir que las máquinas aprendan nuevos conocimientos dotándolas de sentido común y comprensión. Su objetivo en *npl* es inyectar conocimiento lingüístico profundo y estructurado. En el marco de la mayor conferencia de inteligencia artificial de Europa (**ECAI** [14]) se ha organizado un taller [15] para mostrar lo novedoso que es este enfoque y sus posibilidades futuras.

La arqutitectura del clasificador está basada en Perl y su uso es mediante consola de comandos. De esto mismo surge la necesidad de adaptar el clasificador a un entorno web moderno que será el implementado en este proyecto, asegurando su acceso y proporcionando interfaces adaptadas para la visualización de su información por perfiles no tecnológicos.

1.2 Objetivos

El objetivo principal del proyecto es *el desarrollo de una aplicación web para la visualización de datos de bots políticos en Twitter.*

La aplicación permitirá así, persistir datos fuentes desde twitter y los resultados del clasificador asociados a ese input: cuentas, tweets, interacciones, resultados del clasificador, etc. Los usuarios pueden dejar anotaciones asociadas a los conjuntos permitiendo que se comparta información técnica entre los expertos.

En la aplicación web, los usuarios deberán registrarse o iniciar sesión para poder hacer uso del clasificador. El clasificador recibirá como entrada un archivo que contiene las cuentas que van a ser clasificadas junto con un nombre y una descripción que se asociarán al conjunto. Una vez finalizado el proceso, se podrá acceder a las distintas visualizaciones disponibles.

La aplicación cuenta con tres roles diferenciados: investigadores, administradores e invitados.

Los investigadores podrán introducir un fichero de cuentas para clasificar y tendrán acceso a todos los conjuntos del sistema. Además, pueden acceder a toda la gestión relativa a su perfil. Dentro de cada visualización, podrán acceder a la información detallada de cada una de las cuentas y añadir comentarios a los conjuntos que sean de potencial interés para otros expertos. Los administradores tendrán además permiso para eliminar conjuntos y/o notas de expertos.

Los invitados podrán ver todos los conjuntos y acceder a las diferentes visualizaciones pero no podrán hacer uso del clasificador ni añadir comentarios a los conjuntos. Este rol está destinado a usuarios colaboradores en proyectos de investigación o a medios de comunicación colaboradores.

Se han estudiado varias técnicas de visualización de información, asi como sus correspondientes tipologías. Finalmente, se ha optado por la implementación de dos técnicas InfoVis para la inforamación a representar:

- La visualización en *hexagonal grid heat map* [16], [17] busca mostrar una representación general de los resultados del clasificador para ver la incidencia de bots en el conjunto analizado.
- La visualización en *Force-directed graph* [18],[19] pretende facilitar la busqueda de interacciones entre las cuentas del conjunto para así detectar patrones en estas.

Las visualizaciones cuentan con filtros que permiten a los expertos hacer busquedas más concretas dentro del conjunto de datos. Los filtros varían en función de la visualización y los parámetros con mayor valor o influencia.

1.3 Estructura de la memoria

La memoria del trabajo está estructurada en diez capítulos, siendo el primero de ellos esta introducción.

- Segundo capítulo - **Fundamentos tecnológicos y herramientas utilizadas**: Se dará un repaso a todas las tecnologías empleadas en el desarrollo del proyecto y se expondrán los motivos por los que fueron escogidas.

- Tercer capítulo - **Metodología**: Se describirá la metodología de desarrollo seguida y las adaptaciones requeridas para el proyecto.
- Cuarto capítulo - **Análisis**: Se detallarán las funcionalidades de la aplicación, haciendo especial incapié en las funcionalidades específicas para la visualización e interacción con los datos.
- Quinto capítulo - **Diseño**: Descripción de la arquitectura de la aplicación web, tanto del lado servidor como del lado cliente.
- Sexto capítulo - **Planificación y evaluación de costes**: Se detallará la estimación realizada en un primer momento y se comentará el posible coste real de la aplicación web, teniendo en cuenta las horas empleadas y los distintos perfiles adoptados.
- Septimo capítulo - **Implementación**: Explicación del desarrollo técnico del trabajo de manera cronológica.
- Octavo capítulo - **Pruebas**: Se expondrán las pruebas realizadas y sus resultados así como se expondrán los problemas surgidos.
- Noveno capítulo - **Producto final**: Se describirá las funcionalidades y características desarrolladas hasta el momento en la aplicación.
- Decimo capítulo - **Conclusiones**: En el último capítulo se hablará del resultado general obtenido y de las posibles líneas futuras de trabajo.

Capítulo 2

Fundamentos tecnológicos y herramientas utilizadas

PARA el desarrollo del proyecto se han utilizado una serie de tecnologías y herramientas que han facilitado la labor del desarrollo.

En este capítulo se explican brevemente estas tecnologías y herramientas, los motivos por los que se han usado y posibles alternativas.

2.1 Tecnologías

Esta sección se divide en tres partes: Back-End o lado servidor, Front-End o lado del cliente y herramientas de desarrollo.

2.1.1 Backend

El lado del servidor o Back-End se ha construido principalmente con Java EE, MySQL, Hibernate y el framework Spring.

Java

Java es un lenguaje de programación multiplataforma y orientado a objetos desarrollado por James Gosling de Sun Microsystems y publicado en 1995. Su sintáxis deriva de C y C++

aunque dispone de menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones Java son compiladas a bytecode y pueden ser ejecutadas en cualquier máquina virtual Java, independientemente del software o la arquitectura subyacente. Actualmente es uno de los lenguajes más populares y usados del mundo, particularmente para aplicaciones web clientes-servidor.

Como alternativa a Java tenemos .NET de Microsoft, siendo las dos las soluciones más conocidas y populares para el desarrollo de aplicaciones web. La plataforma .NET permite programar en varios lenguajes y cuenta con soporte actualizado por parte de Microsoft.

Los principales motivos para haber escogido java son:

1. El gran soporte por parte de la comunidad del que dispone, haciendo sencillo encontrar soluciones a los problemas que se puedan plantear en el desarrollo
2. Java es el lenguaje más usado durante la carrera, por lo que el dominio sobre este es mucho mayor.
3. Debido a que la capa cliente se desarrollará con tecnologías y librerías basadas en JavaScript, la gran compatibilidad de este lenguaje con Java facilitará la comunicación entre lado servidor y cliente

Se ha usado la JDK 12 (*Java Development kit*).

Mysql

Mysql [20] es un sistema de gestión de base de datos relacional, multihilo y multiusuario basado en SQL. Es el SGBD de código abierto con mayor popularidad, junto a Oracle y Microsoft SQL Server. Utiliza el motor InnoDB.

Otras posibles alternativas son Microsoft SQL Server, MongoDB o PostgresSQL. Las principales razón para haber escogido MySQL son.

1. Mayor familiaridad con la plataforma debido a su uso frecuente en las prácticas de la carrera.
2. Tecnología de código abierto con licencia GPL.
3. Fácil instalación y mantenimiento sin perder por ello profundidad o robustez.
4. Tiene un gran soporte por parte de la comunidad, por lo que ante cualquier problema o cuestión se puede encontrar gran cantidad de documentación.

Se ha utilizado la versión 8.0.19.

Hibernate

Hibernate [21] es una herramienta de mapeo objeto-relacional para el lenguaje de programación Java de código abierto bajo licencia GNU GPL. Esta tecnología facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de datos de una aplicación mediante archivos declarativos (XML) o anotaciones en los beans de las entidades.

Hibernate implementa el estandar JPA, lo que permite construir una capa de persistencia apoyándose en las definiciones y reglas que ofrece la especificación.

JPA resuelve el problema de almacenamiento de objetos en una base de datos relacional, lo que permite correlacionar una tabla con una clase y cada columna contra un atributo de dicha clase. La implementación con JPA se encagará de ocultar la complejidad del acceso a datos exponiendo solamente objetos.

Una posible alternativa podría haber sido JDBC, pero se ha escogido Hibernate por los siguientes motivos.

1. La sencillez y rapidez de trabajar directamente con entidades en lugar de con consultas logrando así una abstraccion del modelo relacional y siguiendo el paradigma de programación orientada a objetos.

2. Mayor dominio y gran popularidad de la tecnología.

Spring

Spring [22] es un framework de código abierto para el desarrollo de aplicaciones para el lenguaje de programación Java. Su principal característica es el contenedor de inversión de control (IoC) que se dedica a instanciar, inicializar y conectar objetos de la aplicación. Los objetos gestionados por Spring se denominan beans. La configuración de estos objetos son conocidos como metadatos de configuración y se puede proporcionar tanto en un archivo de configuración XML o en anotaciones en las clases.

Además, Spring ofrece ventajas como una capa de abstracción de decnologías existentes como servlets o jdbc, soporte de primer nivel para frameworkds de código abierto comunes

como Hibernate, facilidad para la realización de pruebas tanto unitarias como de integración, etc.

Spring tiene un diseño modular, por lo que pueden cargarse únicamente los módulos necesarios reduciendo así el peso de la aplicación.

Como alternativas podríamos destacar Google Web Toolkit (GWT) o JavaServer Facers (JSF). Se ha optado por Spring por los siguientes motivos.

1. Es un Framework modular lo que permite gran personalización y que pueda ser usado tanto en proyectos pequeños como en grandes.
2. Incluye procesos de seguridad configurables que soportan un rango de estándares, protocolos, herramientas y prácticas a través de su módulo Spring Security.
3. Experiencia con el Framework además de contar con una gran popularidad lo que facilita encontrar mucha documentación sobre el mismo y posee un gran soporte de la comunidad.

Se ha usado la versión 2.2.6.

Maven

Maven [23] es una herramienta software para la gestión y construcción de proyectos Java. Utiliza un Project Object Model (POM) en formato XML para describir el proyecto software a construir, indicando sus dependencias de otros módulos, componentes externos, orden de contrucción de sus elementos, etc.

Una característica clave es su capacidad para descargar dinamicamente plugins, tanto del repositorio Maven por defecto como otros repositorios externos que se añadan.

Una alternativa a Maven es Gradle. Esta herramienta introduce un lenguaje específico del dominio basado en Groovy en lugar de utilizar ficheros XML. Gradle está pensado para construcciones multi-proyecto y da soporte a construcciones incrementales determinando

de manera automática que partes del árbol de construcción están actualizadas de modo que cualquier tarea dependiente de aquellas partes no necesitarán ser ejecutadas.

Los principales motivos que han llevado a escoger Maven son:

1. Experiencia con la herramienta además de una gran documentación tanto en internet como en apuntes de distintas asignaturas de la carrera.
2. Automatización del proceso de agregación de dependencias requeridas por el proyecto además de que se encarga de las dependencias transitivas.

Se ha usado su versión 3.6.3.

Junit5

JUnit [24] es un framework para hacer pruebas unitarias de aplicaciones Java. Permite realizar la ejecución de clases Java de manera controlada para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

Si el valor de retorno es el esperado JUnit devolverá que el método pasó exitosamente la prueba, en caso contrario devolverá un fallo en el método correspondiente.

El principal motivo para escoger JUnit es su popularidad y la experiencia con el framework, que ha sido usado durante toda la carrera.

Mockito

Mockito es un framework de código abierto para pruebas unitarias en el lenguaje Java. El framework permite la creación de objetos simulados en las pruebas unitarias para poder comprobar el comportamiento de una clase sin tener en cuenta su integración con otras.

2.1.2 Frontend

El lado del cliente o Front-End se ha construido con la librería React.

HTML

HTML son las siglas de HyperText Markup Language y hace referencia al lenguaje de marcado para la elaboración de páginas web. Define la estructura básica y un código (código HTML) para la definición de contenido de una página web, como texto, imágenes, vídeos, etc.

Es un estándar a cargo del World Wide Web Consortium (W3C), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web. Todos los navegadores actuales han adoptado este estándar por lo que es fundamental en cualquier desarrollo de una aplicación web.

D3.js

D3.js (**Data-Driven Documents**) [25] es una librería JavaScript usada para generar visualizaciones de datos dinámicas e interactivas en navegadores web. Basado en los estándares de gráficos vectoriables escalables (SVG), HTML5 y hojas de estilo en cascada (CSS).

Ha habido numerosos intentos de llevar la visualización de información a las aplicaciones web. Algunos de estos precursores fueron Prefuse, ActionScript y Protovis. Este último, sirvió como experiencia para desarrollar D3.js que proporcionaría un marco más expresivo que su antecesor y se enfocaría al mismo tiempo en los estándares web con un rendimiento mayor.

La principal alternativa a D3.js es Chart.js, una libreria JavaScript de código abierto y la segunda más popular después de d3. Cuenta con una curva de aprendizaje mucho menor pero es menos personalizable en comparación.

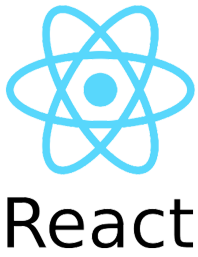
Los motivos para escoger D3.js fueron los siguientes.

1. Una amplia personalización que permite adaptar la visualización a las necesidades del usuario y los datos a representar.

2. Gran popularidad y soporte por parte de la comunidad, especialmente por la comunidad de investigadores, analistas de datos y desarrolladores especializados en InfoVis [26], [27], [28].

3. Especialización en implementaciones centradas en datos (*data-driven*) complejas (grafos, soluciones 3d, etc) en entornos web.

Se ha usado su versión 5.16.0.



React

React [29] es una libreria JavaScript de código abierto basada en el paradigma de programación orientada a componentes y diseñada para crear interfaces de usuario con el objetido de facilitar el desarrollo de SPA (Single-Page Application. Cada componente es una pieza con la que el usuario puede interactuar, estos son reutilizables y se combinan para formar otros componentes mayores hasta formar la página web completa. Cada componente almacena un estado que, al ser modificado, se activará una nueva renderización.

Una SPA es una aplicación web que carga una única página HTML y dinámicamente actualiza esa página mientras el usuario interactua con la aplicación. Las SPAs utilizan AJAX y HTML5 para crear aplicaciones web fluidas y responsive, sin un página constante que se recarga.

React mantiene un virtual DOM propio para determinar que partes han cambiado comparando contenidos entre la nueva versión y la almacenada, utilizando el resultado para determinar como actualizar eficientemente el DOM del navegador.

Las grandes ventajas de usar React para construir el Front-End de nuestra aplicación son:

1. El DOM virtual proporciona fluidez al usuario al navegar por la aplicación web y también rapidez en la carga de las páginas.

2. Gran popularidad y soporte por parte de la comunidad además de experiencia con la libreria.

3. La integración con d3.js, a pesar de ambos querer tomar el control del DOM del navegador, es bastante intuitiva ya que React se hace cargo del DOM mientras que relegamos a d3.js a cargo de una de las ramas del dom, la correspondiente al elemento que ocupe nuestra visualización.

La principal alternativa más popular es Angular, un Framework mantenido por google desarrollado en TypeScript bajo licencia de código abierto. Se descartó por la inexperiencia en el Framework teniendo en cuenta que D3.js ya cuenta con una gran curva de aprendizaje.

Se ha usado su versión 16.12.

Redux

Redux [30] es una librería JavaScript para administrar el estado de la aplicación y se usa frecuentemente junto a librerías o frameworks como React o Angular. Redux es usado mayoritariamente cuando la aplicación a desarrollar tiene una cantidad razonable de datos que cambien frecuentemente pues ayuda a gestionar el estado de distintos componentes.

El principal motivo para haber escogido Redux como librería para gestionar el estado es el gran soporte de la comunidad a su integración con React.

Se ha usado su versión 4.0.5.

Bootstrap

Bootstrap [31] es una librería multiplatafroma de código abierto desarrollada por Twitter para diseño de aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadrados, etc. basados en HTML y CSS además de algunas extensiones opcionales de JavaScript. Las páginas web que usan Boostrap se adaptan automáticamente al espacio disponible del dispositivo desde el que se acceda a la aplicación web, consiguiendo así un diseño responsive.

El principal motivo para usar este framework es la facilidad en su uso y su diseño responsive que permite adaptar nuestro sitio web dinamicamente a la mayoría de pantallas.

Se ha usado su versión 4.4.1.

Yarn

Yarn [32] es un administrador de dependencias enfocado al Front-End, es código abierto y fue creado por miembros de Facebook y google. Sus principales características son.

1. Yarn almacena en caché local todos los paquetes que descarga por primera vez para evitar descargarlos más de una vez. Esto hace que puedas realizar instalaciones aunque no cuentes con conexión a internet.
2. Gracias a la parelización de los procesos de descarga e instalación, Yarn no detiene el proceso de instalación cuando una dependencia falló. Vuelve a realizar las solicitudes necesarias para tener correctamente instalados los paquetes.

Se ha usado su versión 1.22.

2.1.3 Herramientas de desarrollo

Durante el desarollo del proyecto se utilizaron las siguientes herramientas para agilizar el proceso y seguir buenas prácticas.

Eclipse

Eclipse [33] es un entorno de desarollo integrado (IDE) de código abierto multiplataforma. Permite de manera simple y rápida desarrollar nuestros proyectos con múltiples herramien-

tas que facilitan el proceso: Buscador, herramienta de refactorización, generación de código automático, etc.

Algunas de las ventajas que tiene Eclipse y por las cuales se escogió como IDE son.

1. Amplia experiencia con el entorno además de contar con una configuración propia para desarollo de proyectos Java.
2. La gran cantidad de widgets de los que dispone su tienda que permiten personalizar y agilizar procesos como la integración, el control de versiones, la detección de bugs, etc.

Otras alternativas populares son IntelliJ IDEA o NETbeans.

Se ha usado en su versión 4.9 *Foundation*.

GitLab

GitLab [34] es una plataforma de almacenamiento remoto que permite la creación de repositorios manejables y clonables desde cualquier equipo. Permite controlar los cambios realizados sobre el código fuente del proyecto además de revertirlos. Es de código abierto y basado en el sistema de control de versiones Git.

Algunas alternativas a GitLab son GitHub o BitBucket. Se optó por GitLab debido a su facilidad de uso además de ser el sistema del que dispone nuestra universidad.

Draw.io

Draw.io es una herramienta de dibujo que facilita la creación de diagramas UML, modelos E-R, diagramas de flujo, etc. Es de código abierto y utiliza la libreria mxGraph como base.

Hay una gran variedad de herramientas de diagramación, entre ellas algunas basadas en web como LucidChart. Los motivos principales para escoger Draw.io son.

1. Su portabilidad, ya que aunque cuenta con aplicación para varios sistemas operativos, siempre se podrá usar su versión web.
2. Su facilidad de uso.

Se ha usado en su versión 1.13.

SpotBugs

SpotBugs es un programa que realiza análisis estáticos buscando bugs en código Java. Es software libre y es integrable con Eclipse en su versión plugin.

La versión usada por el plugin es la 4.0.

Pencil

Pencil es una herramienta de prototipado de código abierto intuitiva y sencilla.

Se escogió como herramienta por su diseño simple y facilidad de uso. Otras alternativas hubiese sido LucidChart o Balsamiq.

Taiga.io

Taiga [35] es un sistema de gestión de proyectos online, de código abierto y gratuito para desarrollos ágiles. Taiga permite organizar tareas, historias de usuario y sprints de un proyecto. A raíz del desarrollo se van generando gráficos de utilidad sobre trabajo pendiente, ritmo de trabajo, etc.

La principal alternativa para desarollo con metodologías ágiles es Trello. Se optó por taiga por su división natural en sprints.

Visual Studio Code

Visual Studio Code [36] es un editor gratuito de código abierto creado por Microsoft para Windows, Linux y MacOS. Tiene soporte para una variedad de lenguajes de programación, incluidos Java, JavaScript y C++. Incluye soporte para depuración, refactorización de código

y Git integrado. Además se pueden añadir multiples extensiones que permiten añadir distintas funcionalidades adicionales. Se ha usado para desarrollar el Front-End por sus multiples herramientas para tecnologías web tales como React, HTML, CSS, etc.

Una alternativa a Visual Studio Code es Atom, un editor de gran popularidad pero que no proporciona algunas de las funcionalidades de Visual Studio sin el uso de extensiones.

Se ha usado su versión 1.46.

Overleaf y LaTeX

LaTeX [37] es una herramienta de software libre para la composición de textos, especialmente a creación de libros y documentos científicos y/o técnicos. Overleaf es un editor colabroativo de LaTeX basado en la nube que se utiliza para escribir, editar y publicar documentos científicos.

El mayor inconveniente de LaTeX frente a cualquier otra herramienta de composición de documentos es su curva de aprendizaje.

Se ha usado esta herramienta para la redacción de la memoria del proyecto.

2.2 Hardware

Los recursos hardware y de servicios utilziados para el desarrollo del proyecto son:

- Ordenador de sobremesa con Intel Core i7-6700K, 8gb de RAM 256GB SDD y 1TB HDD.

- Acceso a internet de banda ancha
- Xiaomi Mi Notebook pro Intel Core i5-8250U 8gb RAM 256GB SSD

Capítulo 3

Metodología

En este capítulo se introduce la metodología de trabajo seguida en el desarollo del proyecto, que es una adaptación de la metodología ágil Scrum [38].

3.1 Scrum

Scrum es una marco de trabajo para desarollo ágil e incremental que define unos roles con distintas responsabilidades y tareas, una estrategia de desarollo incremental basado en *sprints* (ciclos cortos de desarollo) y una serie de artefactos.

Scrum se caracteriza por la flexibilidad durante el desarrollo centrándose en la capacidad del equipo de responder a los cambios (frecuentes en desarrollo software).

Al tratarse de un equipo de desarollo de una persona la metodología ha sido adaptada centrándose en el flujo de trabajo. Las principales ventajas de aplicar esta metodología al desarrollo del proyecto son:

- El dividir el desarrollo del proyecto en entregas parciales y regulares, se flexibiliza el entorno haciendo mucho más sencilla la entrada de cambios en los requisitos.
- El desarrollo por sprints permite conocer la velocidad media del equipo, con lo que es posible estimar de manera mucho más precisa cuanto puede llevar una historia de usuario de tal manera que los sprints estén bien acotados.
- La focalización en ciertos requisitos en un sprint, permite centrar la atención y ser más

productivo durante el desarrollo del proyecto. El trabajo paralelo en varias funcionalidades suele ser más lento y además deriva en un software de peor calidad.

- Al eliminar Scrum la documentación excesiva que en las metodologías clásicas es obligatoria y que haría muy pesada y complicada la realización de un proyecto de este estilo por un equipo de una persona, permite centrar el esfuerzo en el desarrollo del proyecto, obteniendo un mayor beneficio.

- Debido a la falta de experiencia con algunas de las tecnologías usadas es difícil estimar el alcance del proyecto. Por ello, permite centrarse en las historias de usuario fundamentales y una vez se sepa el ritmo de trabajo del equipo, evaluar si sería posible llegar a ciertas historias de usuario más superfluas.

- Los potenciales usuarios de la aplicación tienen perfiles no tecnológicos, muy enfocados en disciplinas humanísticas. Scrum permite centrarse en una toma ágil de requisitos priorizando darle valor añadido a los usuarios. Al priorizar ese valor añadido y centrarlo en las preguntas de investigación e intereses en los datos de los usuarios, conseguimos una aplicación más real y valiosa para ellos.

3.1.1 Roles

En Scrum existen una variedad de roles especializados de los que podemos destacar los siguientes:

- ***Product Owner***: Representa a los interesados (*stakeholders*) y es el responsable de que el equipo entienda los elementos del *Product Backlog*. Se encarga de su gestión, escribiendo las historias de usuario y ordenandolas de la mejor manera posible para el desarrollo. Es un rol que actua de intermediario entre la organización y el equipo de desarrollo buscando cumplir con los objetivos.

- **Scrum Master**: Es el responsable de eliminar los obstáculos que impiden que el equipo alcance el objetivo del *sprint*. El *Scrum Master* no es el lider del equipo ya que este está auto-organizado, pero se encarga de hacer cumplir las pautas de la metodología como es debido.

- **Equipo de desarrollo o development-team**: El equipo de desarollo tiene la responsabilidad de entregar el producto. Es recomendable un pequeño equipo multifuncional de 3 a 9 personas con las habilidades necesarias para entregar un Incremento. Está auto-organizado y auto-gestionado.

Hay otros roles auxiliares que no se involucran directamente en el proceso Scrum pero participan y forman parte del desarrollo del proyecto.

- **Usuarios**: Destinatario final del producto y quien lo va a usar.
- **Stakelholders/cliente**: Son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Solo participan directamente durante las revisiones del sprint.

3.1.2 Artefactos

Los artefactos Scrum son los elementos que garantizan la transparencia y el registro de la información fundamental del proceso. Deben estar diseñados para maximizar el entendimiento de la información clave, es decir, deben utilizar un lenguaje de negocio.

- **Product Backlog**: Se trata como un documento de alto nivel para todo el proyecto. Es el conjunto de todos los requisitos del proyecto priorizados según su ROI (*Retorno sobre la inversión*). Solo puede ser modificado por el *Product Owner* y contiene estimaciones realizadas a grandes rasgos para poder escoger que tareas se podrían completar en tiempo.
  - **Historia de usuario**: on los requisitos del proyecto expresados en lenguaje de negocio desde el punto de vista del usuario/cliente. El *Product Owner* es el encargado de su redacción y son la unidad básica de un *sprint*.
  - **Épicas**: Historias de usuario de gran tamaño que pueden y deben ser subdivididas en pequeñas historias antes de ingresar a un *Sprint Backlog*.
- ***Sprint Backlog***: Es el subconjunto de historias de usuario que serán desarrolladas durante el siguiente sprint. Una vez introducidas las historias de usuario en el Sprint Backlog, se describe cómo el equipo va a implementar los requisitos durante el sprint. Se dividen en tareas que deben de tener una duración menor a dieciseis horas (en caso contrario la tarea debe ser dividida en subtareas). Estas tareas no deben ser asignadas si no que los miembros del equipo deben de tomarlas del modo que les parezca más adecuado.
  - **Tarea**: Son divisiones de las historias de usuario que se expresan en lenguaje de dominio técnico, propio del equipo de desarrollo.

- ***Burn-Down* chart**: Es un gráfico diseñado para ayudar al equipo a monitorizar su progreso y para servir de indicador sobre sus posibilidades de completar lo planificado al finalizar el sprint. Cada historia de usuario tendrá unos puntos en función de su complejidad y la gráfica medirá cuantos puntos restan por completar y el tiempo disponible hasta finalizar el sprint.

3.1.3 Eventos de Scrum

Los eventos forman una parte fundamental de la metodología (Figura 3.1), ya que crean un patrón constante y minimizan la necesidad de reuniones no definidas permitiendo a los involucrados centrarse en el desarollo. Los eventos tienen un tiempo máximo definido (*time-boxing*) pudiendo acabarse siempre que se logre el objetivo del evento. Cada evento establece formas para la inspección y adaptación de algún aspecto del proyecto.

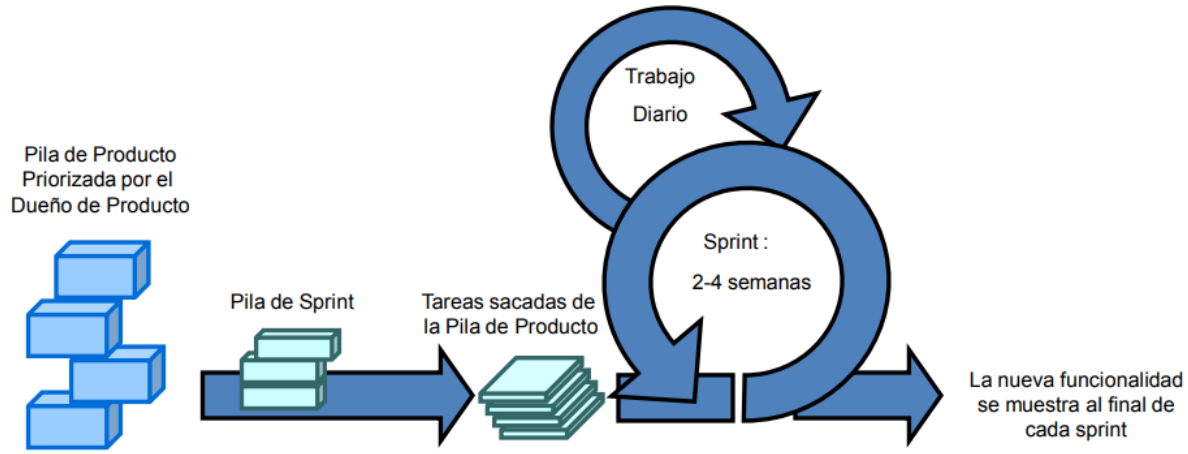


Figura 3.1: Ciclo básico de desarrollo Scrum. *Fuente: Adapted from Agile Software Development with Scrum by Ken Schwaber and Mike Beedle.*

Sprint

Es un período en el cual se lleva a cabo el desarrollo del proyecto. Es recomendado que la duración sea constante y definida por el equipo con base en su propia experiencia y ritmo de trabajo. Se puede comenzar con una duración concreta (tres semanas) y ajustar los siguientes en base al resultado de este.

Como ya mencionamos antes, un sprint debe de aportar valor al usuario final. Por ello, se recomienda no agregar objetivos al sprint a menos que su falta amenace el éxito del proyecto. Esta constancia permite concentrarse y tener claro los objetivos a alcanzar. Por lo general, la duración de un sprint puede variar entre dos y cuatro semanas.

Solo en situaciones excepcionales el cliente o el equipo puede solicitar una terminación anormal de la iteración, por ejemplo en contextos en los que el proyecto haya cambiado o si el equipo encuentra imposible terminar con la previsión de objetivos planteada.

Durante el ciclo de trabajo se realizan varios tipos de reuniones para hacer seguimiento continuo del proyecto, facilitar la comunicación del equipo y adaptar el proyecto a posibles nuevos requisitos.

Sprint planning

El sprint planning es un evento previo a cualquier *sprint* en el cual se seleccionan las historias de usuario del *Product Backlog* que se van a realizar durante el *sprint*. Debe hacerse una estimación de complejidad y esfuerzo para cada historia de usuario para poder organizar de forma correcta el *sprint*, además de atender a la prioridad que ha establecido en las historias el *Product Owner*. Una vez seleccionadas, los miembros del equipo harán una división de las historias en tareas con lenguaje técnico, aléjandose de la visión de usuario propia de las historias de usuario.

Daily Scrum

Reunión diaria del equipo para hacer puesta en común de lo hecho durante el día y de lo que se tiene pensado hacer al día siguiente. En este evento también se tratan los problemas que cualquier miembro del equipo pudiese tener y el resto deberá preguntarse en qué puede ayudarlo y en cómo solucionar el problema. En la reunión participará todo el equipo y puede aparecer también la figura del *Product Owner*. Debe ser breve (alrededor de unos diez minutos) y por eso es frecuente realizar la reunión de pie.

Sprint Review

Es la primera reunión celebrada al final del *sprint*. En ella se exponen las tareas realizadas durante el *sprint* para su revisión. El *Product Owner* y los stackholders erán los que finalmente

decidan si fueron completados los objetivos finales del *sprint*. Los elementos no finalizados se deben poner de nuevo en el *Product Backlog* y si hiciera falta, calcular de nuevo su prioridad o su complejidad.

Retrospectiva

Al final de cada *sprint*, se lleva a cabo una reunión en la cual todos los miembros del equipo dejan impresiones sobre el *sprint* recién superado. Su duración es de entre una y tres horas. Se usa principalmente para detectar debilidades del equipo y para ser consciente de las fortalezas, para poder apoyarse en ellas. A pesar de coincidir cronologicamente con la *sprint* Review, sus objetivos son muy distintos y por lo tanto deben tratarse por separado.

3.2 Adaptación

La metodología de desarollo adoptada en este proyecto ha sido una adaptación de la metodología Scrum descrita en la sección anterior. Se van a detallar a continuación las modificaciones hechas en el proceso Scrum y los motivos para estas modificaciones.

1. **Roles**: Debido a las circunstancias del proyecto, todas las responsabilidades de todos los roles del equipo de desarrollo recaen sobre la única persona que conformará el equipo, el autor del trabajo. Si bien el equipo se completa con la supervisión del director del trabajo, que también asumirá en ocasiones los roles de cliente y usuario final, proporcionando además acceso a usuarios finales reales en algunos casos.

2. **Reuniones**:

   - ***Sprint Planning* y *Sprint Review***: Reuniones de entre treinta minutos y una hora llevada a cabo al final de un *sprint* para analizar el trabajo realizado y planificar el siguiente.
   - ***Daily Scrum***: Auto-análisis del trabajo realizado el día anterior.

   Además de estas reuniones, la comunicación entre los involucrados en el proyecto siempre fue fluida mediante correo electrónico.

3. ***Sprint***: La metodología Scrum recomienda que la duración de los *sprints* sea constante y el peso de las historias añadidas al *Sprint Backlog* sea compensado. En el desarrollo de

este proyecto la dedicación del equipo ha sido variable a lo largo de los *sprints*. Scrum permite adaptar el proyecto a estas circunstancias.

4. **Historias de usuario**: A pesar de que las historias de usuario deben de ser redactadas por el *Product Owner* desde el punto de vista de valor para el usuario, en los primeros *sprint*s toman un caracter formativo en las nuevas tecnologías necesarias para la realización del proyecto.

La metodología Scrum ha sido adaptada para un entorno de investigación (el proyecto está enmarcado en un proyecto marco de investigación interuniversitaria). Las adaptaciones necesarias se han centrado sobre todo en orientar las historias de usuario y los artefactos Scrum a las preguntas de investigación y necesidades de los usuarios finales.

Capítulo 4

Análisis

En este capítulo se describen los requisitos y necesidades que dan lugar al proyecto, así como su ámbito y modelo conceptual.

El análisis de requisitos se lleva acabo en la parte inicial de un proyecto y es una de las partes fundamentales del desarrollo software. En esta fase se determinarán necesidades o condiciones clave que debe cumplir nuestra aplicación. Antes de iniciar la fase de diseño es muy importante analizar con profundidad el alcance del sistema y sus funcionalidades.

4.1 Ambito de la aplicación y descripción general

La aplicación web desarrollada analizará y persistirá un conjunto de cuentas cada vez que se introduzca un fichero de cuentas en el clasificador. Estes conjuntos tendrán que llevar un nombre y una descripción añadida por el autor del mismo. Cada uno de los conjuntos tendrá disponibles las visualizaciones implementadas.

Los conjuntos de cuentas están formados por cuentas de *Twitter*. Cada una de estas cuentas tiene como mínimo su nombre de usuario, identificador de cada una de ellas y uno o más *Tweets*. Opcionalmente puede contar con biografía, fecha de creación de cuenta, fecha de cumpleaños, ubicación, usuarios seguidores, usuarios siguiendo, etc. Como resultado de clasificar estas cuentas y sus tweets obtendremos un porcentaje de bot o de humano por cada cuenta.

La información relativa a las cuentas y al conjunto, poniendo el foco en el resultado, serán la clave de las visualizaciones.

4.2 Requisitos

La tarea principal del análisis de este proyecto es la recogida de requisitos, parte fundamental pues será la forma de comprender la aplicación que se intenta construir, poniendo en contexto tanto a desarrolladores como a clientes y/o usuarios.

4.2.1 Actores

A la hora de recoger los requisitos se identificaron los siguientes actores que dispondran de distintas funcionalidades:

- **Usuario no identificado**: Usuario visitante de la web que no ha iniciado sesión
- **Usuario identificado**: Usuario que ha iniciado sesión de manera exitosa
- **Administrador**: Usuario encargado de la gestión de la aplicación
- **Usuario Invitado**: Usuario identificado con permisos reducidos.

Todos los actores heredan funcionalidades del actor genérico usuario de la aplicación. A su vez, administrador hereda de usuario identificado y este hereda del actor usuario invitado.

4.2.2 Historias de usuario

Tras un estudio del dominio de la aplicación y de las necesidades de los usuarios, se elaboró el *Product Backlog* de Scrum. A medida que avanzó el desarrollo, algunas de las historias de usuario del *Product Backlog* fueron modificadas, añadidas o eliminadas.

Las historias de usuario son los requisitos de la aplicación que describen características con las que debe contar el sistema. Deben de ser comprensibles para todos los miembros del equipo y bien delimitadas pues van a ser implementadas en la duración de un Sprint.

Teniendo en cuenta el caracter del proyecto, gran parte de las historias de usuario conllevan un estudio de la visualización, teniendo que analizar la información que se expone, como se expone y como puede interactuar el usuario con ella.

Cada historia de usuario contará con un diagrama de casos de uso para aumentar la legibilidad de la memoria. Para el resto de la memoria la unidad de trabajo seguirá siendo la

historia de usuario, evitando entrar en la dicotomía metodológica historia de usuario vs. caso de uso [39].

A continuación se van a detallar cada una de las historias de usuario:

Historia de usuario # 1: Gestión de usuarios

Como usuario de la aplicación quiero poder crear, gestionar y usar mi cuenta para hacer uso de las funciones de la aplicación. 4.1

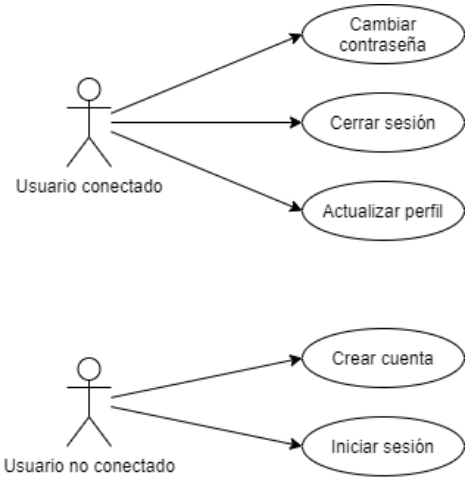


Figura 4.1: Casos de uso de la historia de usuario número #1

Esta historia de usuario implica toda la gestión de usuario de la aplicación. Las funciones a las que hace referencia son:

1. **Crear cuenta**: Permite registrarse en la aplicación para hacer uso de las funcionalidades de un usuario identificado.
   - Notas:
     - Nombre de usuario, nombre, primer apellido, contraseña y correo electrónico obligatorios.
     - El nombre de usuario debe ser único.
2. **Iniciar sesión**: Acceso a la aplicación como usuario identificado con nombre de usuario y contraseña.
3. **Actualizar perfil**: Permite cambiar los datos de perfil: Nombre, apellidos o correo electrónico.

4. **Cerrar sesión**: Cerrar la sesión del usuario en la aplicación para que otra persona en su dispositivo no pueda acceder a la aplicación sin identificarse.

5. **Cambiar contraseña**: Permite cambiar la contraseña del usuario.

Historia de usuario # 2: Uso del clasificador de bots

Como usuario identificado quiero poder añadir nuevos conjuntos de cuentas para poder clasificarlas y visualizar la información. 4.2

Esta historia de usuario hace referencia a la necesidad de embeber el clasificador en nuestra aplicación web. Las funciones a las que hace referencia son:

1. **Subir fichero**: Permite subir el fichero de cuentas y *Tweets* para ser clasificado.

   - Notas:
     - El fichero debe estar en formato .xml

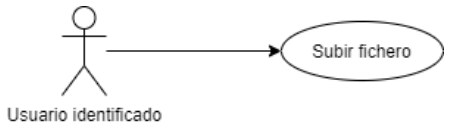


Figura 4.2: Casos de uso de la historia de usuario número #2

Historia de usuario # 3: Visualización resultado clasificador

Como usuario identificado quiero ver una representación de las cuentas de twitter para diferenciar entre las clasificadas como bots y las clasificadas como humanos

Esta historia de usuario se basa en el análisis de las necesidades de visualización de la información y en su implementación. Las funciones a las que hace referencia son:

1. **Ver detalles de un conjunto**: Permite acceder y ver los detalles de uno de los conjuntos añadidos a la aplicación.

2. **Visualización de cuentas en red hexágonal**: Acceso a la visualización de las cuentas de un conjunto ordenadas de más bot a más humano permitiendo ver los detalles de cada una de las cuentas. Se ha usado el patrón *focus + context* de InfoVis para centrar la atención del usuario en sus intereses y dividir la información en capas [40].

(a) **Interacción con los hexágonos**: Al colocar el puntero en uno de los hexágonos este se resalta y se muestra información detalla de la cuenta que representa.

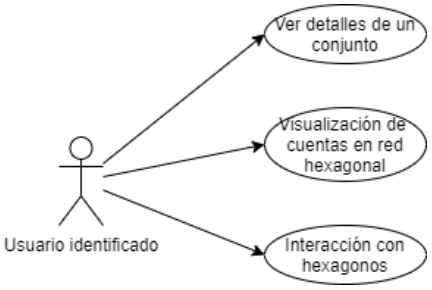


Figura 4.3: Casos de uso de la historia de usuario número #3

Historia de usuario # 4: Filtro de cuentas

Como usuario identificado quiero filtrar las cuentas introducidas según su porcentaje de bot o humano para poder buscar las cuentas con características especificas 4.4.

Las funciones a las que hace referencia son:

1. **Filtrar cuentas por porcentaje**: Se tiene acceso a un menú que permite escoger el porcentaje mínimo de las cuentas.
2. **Filtrar cuentas por tipo**: Se tiene acceso a un menú que permite escoger el tipo de cuenta que se desea.

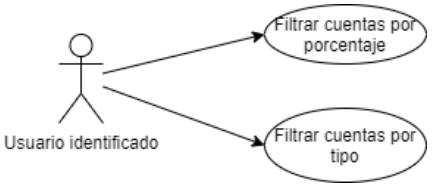


Figura 4.4: Casos de uso de la historia de usuario número #4

4.2.3 Historia de usuario # 5: Gestión de roles

Como administrador quiero poder dar y retirar permisos para gestionar a los usuarios en la aplicación 4.5.

Las funciones a las que hace referencia son:

1. **Cambiar rol de una cuenta**: Cambio de permisos de una cuenta.
2. **Restringir acceso a una cuenta**: Vetar el acceso a un usuario a la aplicación.

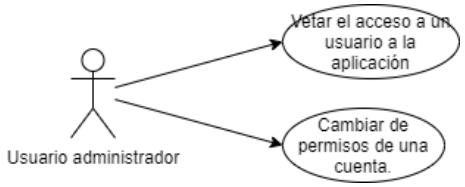


Figura 4.5: Casos de uso de la historia de usuario número #5

Historia de usuario # 6: Visualización de grafo

Como usuario identificado quiero poder ver grafos de relaciones entre las cuentas para poder detectar redes de bots 4.6.

Las funciones a las que hace referencia son:

1. **Visualización de grafo**: Acceso a la visualización basada en nodos y enlaces que representan las interacciones entre las cuentas (*Retweets*, *me gusta* o *seguimiento*) representadas con distintos colores para diferenciarlas.
2. **Interacción con los nodos**: Interacción con un nodo de tal forma que se centre la atención en ese nodo y sus relaciones.

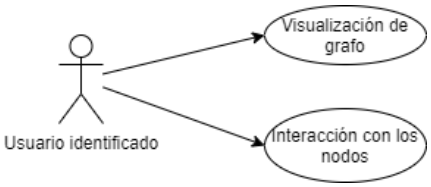


Figura 4.6: Casos de uso de la historia de usuario número #6

4.2.4 Historia de usuario # 7: Detalles de cuenta

Como usuario identificado quiero poder consultar los detalles de una cuenta de twitter introducida en el sistema para poder consultar sus datos o tweets guardados en el sistema 4.7.

Las funciones a las que hace referencia son:

1. **Ver detalles de una cuenta**: Ver detalles de una cuenta: biografía, seguidores, Tweets, distintos resultados del clasificador, etc.

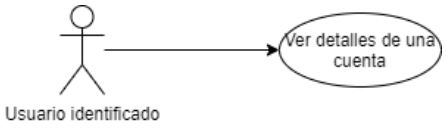


Figura 4.7: Casos de uso de la historia de usuario número #7

4.3 Modelo de datos

En la figura 4.8 se muestra el modelo de datos del dominio (modelo conceptual) sin tener en cuenta especificaciones técnicas.

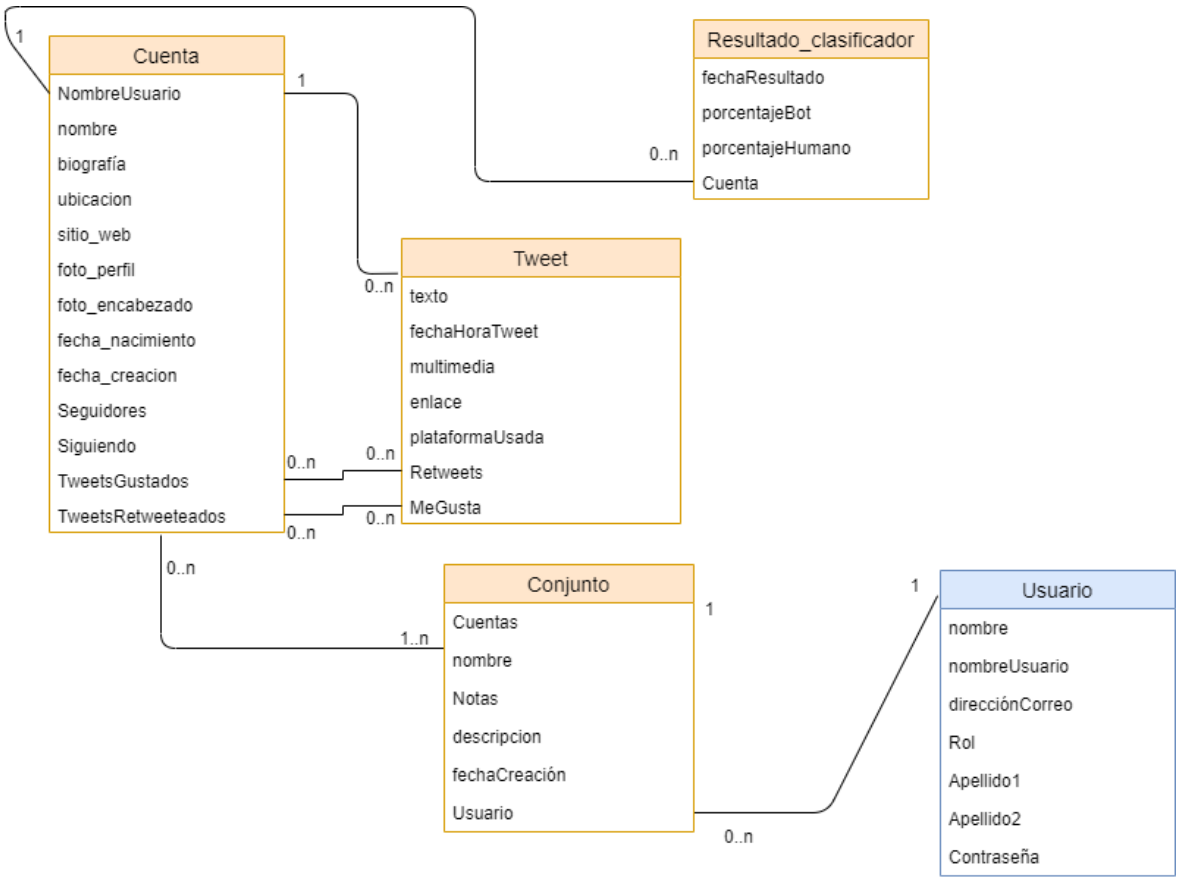


Figura 4.8: Modelo de datos sin especificaciones técnicas

Al seleccionar para persistencia de datos un modelo relacional, se diseñó el diagrama

entidad relación 4.9 que implementa el modelo de datos en el esquema relacional.

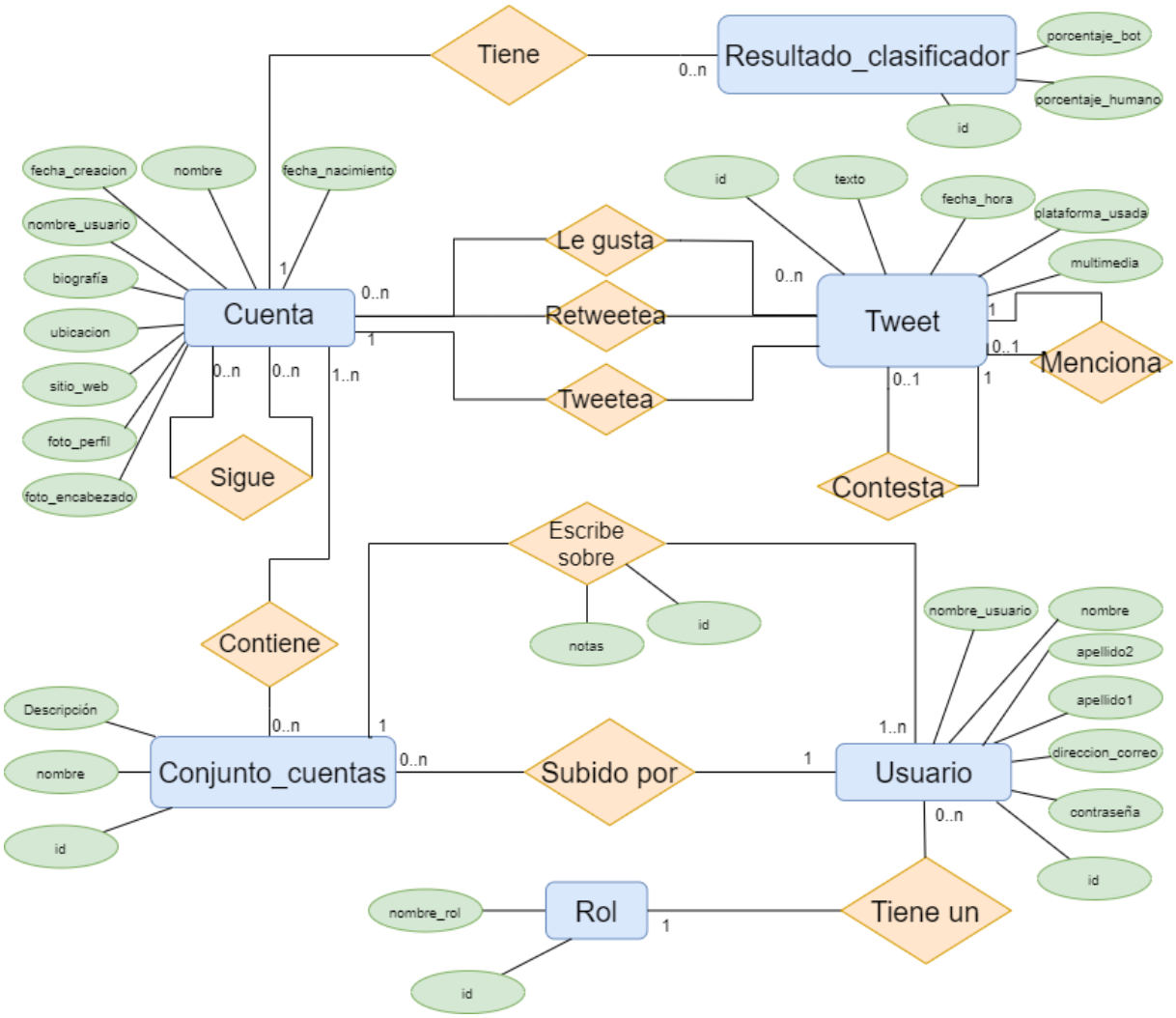


Figura 4.9: Diagrama entidad-relación de la aplicación

A continuación se describirán las entidades y atributos del modelo de datos para poder comprender el dominio.

- **Cuenta**. Representa a un usuario de la red social *Twitter*.
  - *NombreUsuario*: Identificador único de una cuenta en la red social *Twitter*.
  - *Nombre*: Nombre del usuario de la cuenta.
  - *Biografia*: Descripción de la cuenta mostrada en el perfil de la cuenta.
  - *Ubicación*: Ubicación de la cuenta mostrada en el perfil de la cuenta.

  - *SitioWeb*: Enlace a una página web escogida por el usuario de la cuenta mostrada en el perfil.
  - *FotoPerfil*: La foto de perfil de la cuenta.
  - *FotoEncabezado*: Foto de fondo que tiene el perfil.
  - *fechaNacimiento*: Fecha de nacimiento del usuario de la cuenta.
  - *fechaCreación*: Fecha de creación de la cuenta.
  - *seguidores*: Otros usuarios que siguen a esta cuenta.
  - *siguiendo*: Cuentas a las que sigue el usuario de la cuenta.
  - *TweetsGustados*: Tweets con los que ha interactuado dando un me gusta.
  - *TweetsRetweeteados*: Tweets con los que ha interacuado dando *retweet*.

- **Tweet**. Es el elemento usado para compartir un mensaje en *Twitter*.
  - *texto*: Texto del Tweet.
  - *FechaHoraTweet*: Fecha y hora de publicación del *Tweet*.
  - *Multimedia*: Archivo que puede ir acompañado al *Tweet*
  - *Enlace*: Enlace al *Tweet*.
  - *plataformaUsada*: Plataforma desde la que se ha publicado el Tweet (Por ejemplo, móvil *android*, *Iphone*, etc.).
  - *Retweets*: Interacción de otro usuario que hace que se muestre el *Tweet* a los seguidores de la cuenta que lo ha hecho.
  - *MeGusta*: Interacción de otro usuario con el *Tweet*.

- **ResultadoClasificador**. Resultado del clasificador para una cuenta.
  - *fechaResultado*: Fecha en la que se ha clasificado la cuenta.
  - *porcentajeBot*: Probabilidad de que la cuenta sea un bot.
  - *porcentajeHumano*: Probabilidad de que la cuenta sea humana.
  - *Cuenta*: Cuenta de la que se ha obtenido el resultado.

- **Conjunto**. Conjunto de cuentas introducido en la aplicación para ser clasificados.
  - *Notas*: Notas introducidor por los usuarios de la aplicación sobre el conjunto.
  - *Nombre*: Nombre del conjunto.
  - *Cuentas*: Cuentas introducidas .
  - *Usuario*: Usuario que ha introducido el conjunto de cuentas.

  - *fechaCreación*: Fecha de creación del conjunto.
  - *Descripción*: Descripción del conjunto de cuentas.

- **Usuarios**. Usuarios de la aplicación desarrollada.
  - *Nombre*: Nombre del usuario.
  - *NombreUsuario*: Nombre identificador del usuario.
  - *DirecciónCorreo*: Correo electrónico del usuario.
  - *Apellido1*: Primer apellido del usuario.
  - *Apellido2*: Segundo apellido del usuario.
  - *Rol*: Rol del usuario en la aplicación.
  - *Contraseña*: Contraseña del usuario.

Posteriormente fue necesario hacer una adaptación técnica en el modelo de datos incluyendo entidades para asegurar la eficiencia de la aplicación. Dicha adaptación es explicada en detalle en el capítulo 7: Implementación.

Capítulo 5

Diseño

En este capítulo se comentan las decisiones que se han tomado en cuanto al diseño de la aplicación y su arquitectura.

5.1 Maquetas

Se empezó creando maquetas de las visualizaciones y su interfaz de usuario. El objetivo es crear una interfaz simple e intuitiva siguiendo las pautas de *User Interface* y *User Experience* [41].

User Interface hace referencia a la apariencia de la interfaz del software o dispostivo con la que interactuan los usuarios, mientras que *User Experience* es la sensación general del usuario al interactuar con dicha interfaz.

Las maquetas se fueron diseñando a medida que se avanzaba en el desarrollo del proyecto. Lo que buscan las interfaces diseñadas es encontrar la manera más intuitiva y sencilla de mostrar una gran cantidad de información, por ello se han hecho maquetas de diseño centrándose en las funcionalidades clave de las historias de usuarios definidas, especialmente en las de visualización de información. Se ha buscado que la *User Experience* satisfaga criterios estéticos pero principalmente criterios basados en las necesidades de investigación de los usuarios. Las visualizaciones buscan responder a preguntas concretas que los usuarios finales se hacen sobre los conjuntos de datos. Esto convierte el diseño de interfaz en un diseño *data-driven.*

En la disciplina de visualización de la información, como muchas otras áreas de experiencia del usuario es un área subjetiva. Con todo, hay autoridades reconocidas como el estadístico

estadounidense **Edward Tufte** [42], que establecen ciertas pautas a seguir.

1. Excelencia gráfica
2. Integridad visual
3. Maximización de la relación datos-tinta
4. Elegancia estética

Estas pautas y criterios persiguen que la representación de la información se realice de la forma más adecuada al dominio, de manera que el usuario pueda sacar el máximo provecho a la representación.

En base a esto, se ha optado por la visualización en red hexagonal (*hexagonal grid*) pues permite mostrar gran cantidad de información granular (cuentas de *Twitter*) atendiendo a un criterio gradual (el porcentaje de bot o humano) de manera muy intuitiva. Como muestra la figura 5.1 cada uno de los hexágonos representa una cuenta, los colores y su intensidad deben de representar con claridad que tipo de cuenta es cada hexágono y con que certeza (%) lo es.

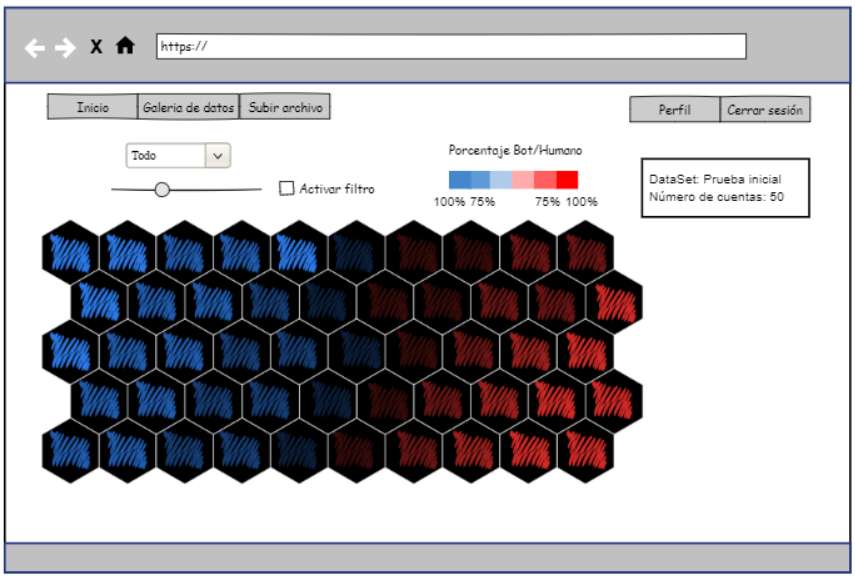


Figura 5.1: Visualización red hexagonal

Aplicando el patrón *focus+context* (Figura 5.2), la interacción debe ser un elemento más de la visualización y por ello y como parte del valor añadido al usuario, interactuar con cada

uno de los hexágonos que representa una cuenta con un resultado específico debe aportar información sobre ella: Nombre, número de *Tweets*, etc. Además, el hexágono seleccionado debe de aumentar su tamaño de tal forma que represente de qué cuenta entre todas las del conjunto estamos viendo la información.

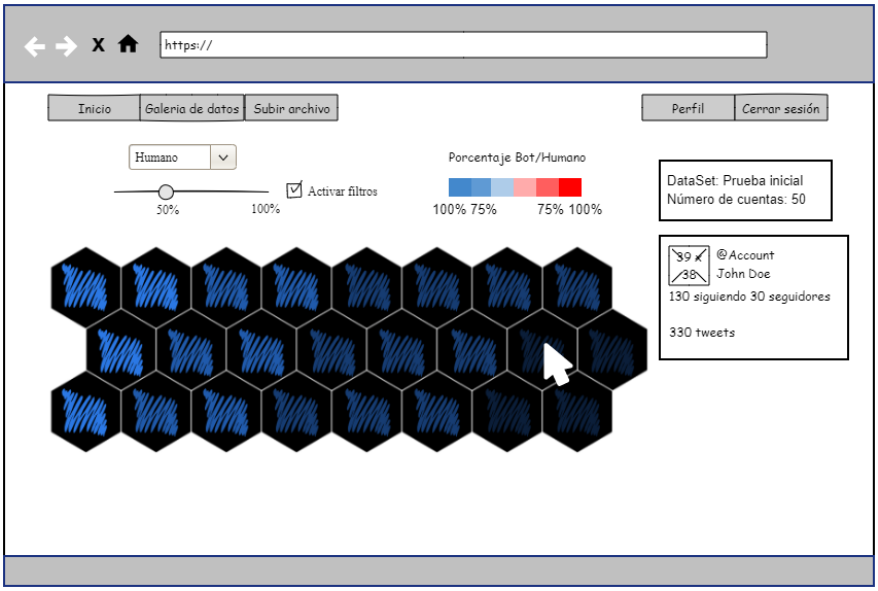


Figura 5.2: Visualización red hexagonal al interactuar con un hexágono

La visualización de grafos (Figura 5.3) por otra parte, se centra en la interacciones entre las cuentas. Surge de la necesidad de los investigadores, no solo de conocer qué cuentas son bots o humanos y en qué porcentaje, sino la influencia a nivel de red de dichas cuentas: si forman *clusters* entre ellas o mantienen interacciones entre ellas dentro de la red social. Estas conexiones son importantes para los investigadores ya que pueden implicar cierta relación en la vida real: mismos usuarios con misma ideología, mismos usuarios detrás de mismo tipo de bots, etc. Es una visualización de grafo cuyos nodos son las cuentas y enlaces sus interacciones.

Como en el caso anterior, en esta visualización cada uno de los elementos visuales representa un tipo de dato. En este caso los colores de los enlaces representan el tipo de interacción entre las cuentas (*Seguimiento*, *me gustas* o *Retweets*) y el tamaño representa la cantidad de interacciones de cada tipo que hay.

Esta visualización (Figura 5.4) también contará con interacción que consistirá en poder seleccionar un nodo de manera que se realice un *zoom in* mostrando información relativa a los enlaces (patrón *focus + context*).

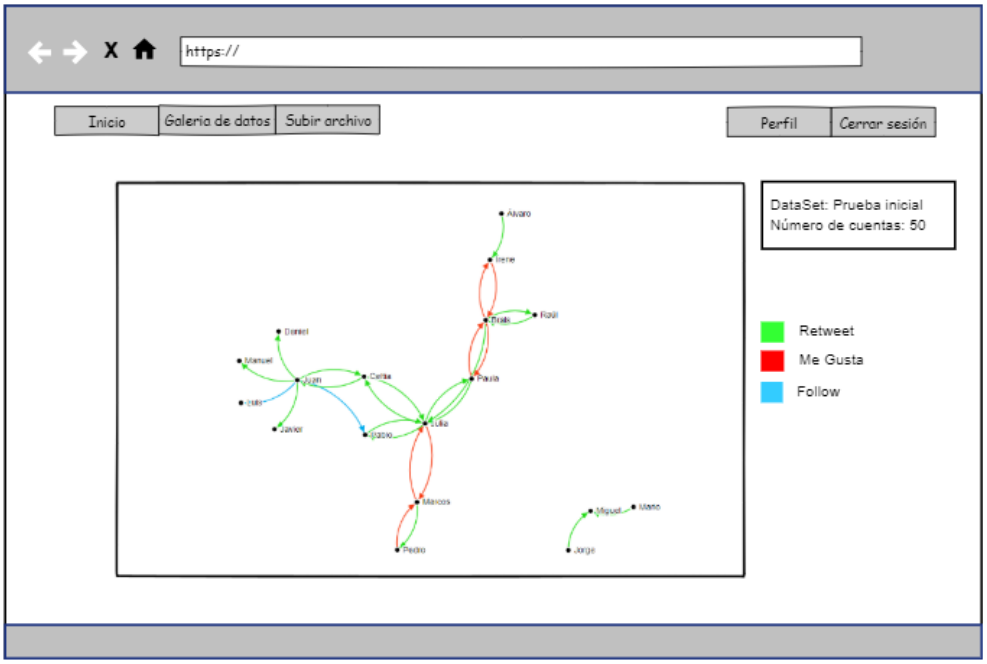


Figura 5.3: Visualización de grafo

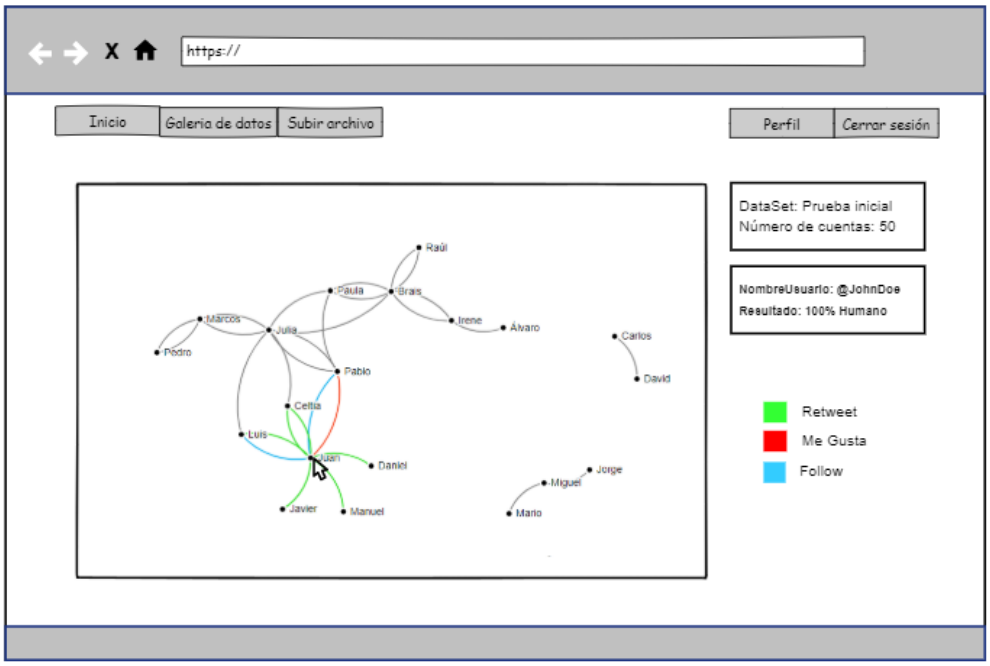


Figura 5.4: Visualización de grafo al interactuar con un nodo

La sencillez es un elemento fundamental en la experiencia de usuario. Siguiendo esa premisa y priorizando las visualizaciones, se decidió que el resto del diseño de la aplicación web mantuviera un estilo de interfaz minimalista.

El acceso a los conjuntos subidos por otros usuarios consiste en una lista con los conjuntos ordenados por la fecha de creación y paginados (Figura 5.5)

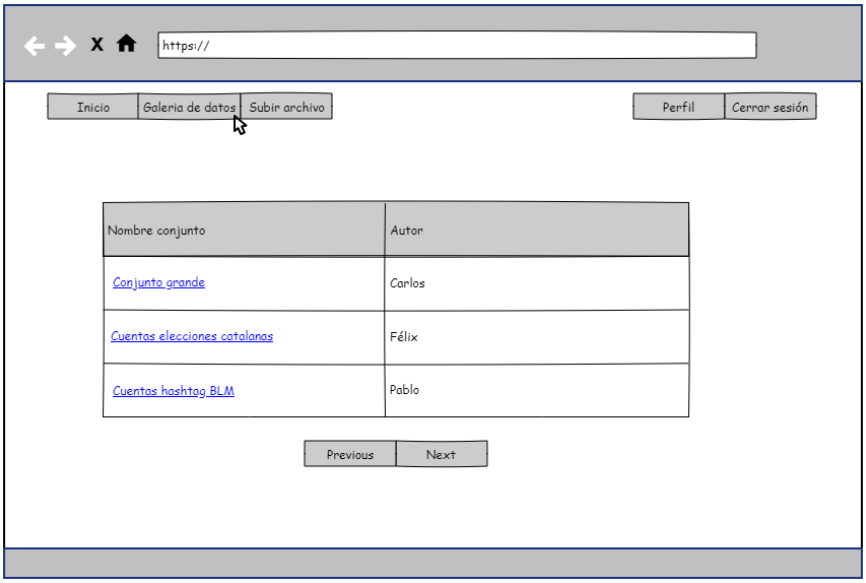


Figura 5.5: Conjuntos listados con enlaces a sus detalles.

Interactuando con el nombre de un conjunto concreto accedemos a sus detalles (Figura 5.6), y a partir de ahí podemos decidir escoger una de las visualizaciones disponibles en el desplegable.

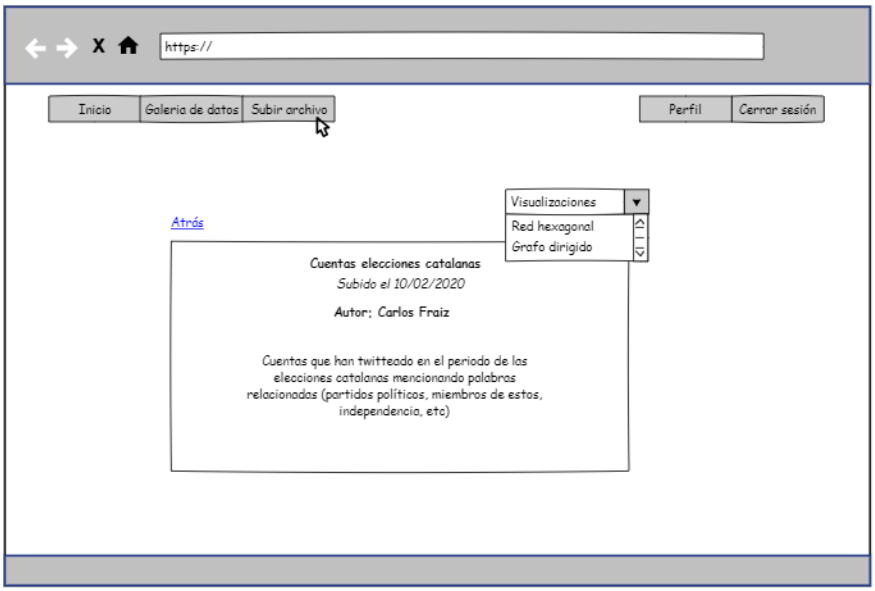


Figura 5.6: Maqueta de la interfaz para ver detalles de un conjunto

Para poder clasificar un conjunto de cuentas es necesario crear un conjunto. Se subirá un fichero de cuentas y se añadirá un nombre y una descripción para el conjunto creado (Figura 5.7).

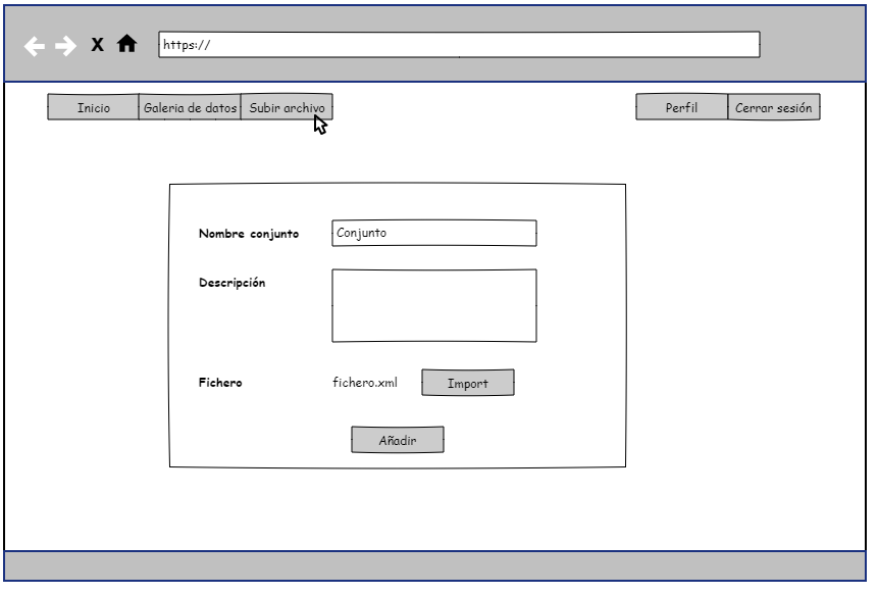


Figura 5.7: Maqueta de la interfaz para añadir un conjunto

5.2 Arquitectura global

El proyecto sigue la arquitectura clásica cliente-servidor por capas (Figura 5.8). El cliente está conformado por la capa de interfaz de usuario y el servidor sigue también una arquitectura en capas.

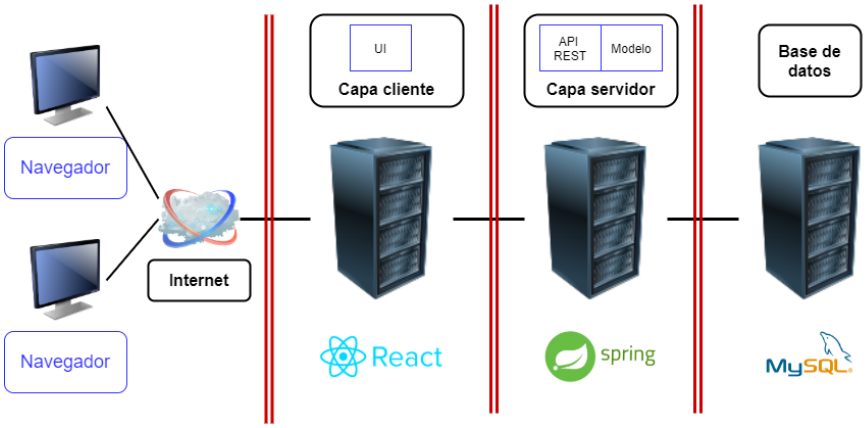


Figura 5.8: Diagrama de arquitectura cliente-servidor por capas

5.2.1 Arquitectura de lado servidor

El Back-End o lado del servidor se divide en las capas web y modelo. Esta última está formada por la capa de lógica de negocio, la capa de dominio y la capa de acceso a datos.

La capa web consiste en la API REST de la aplicación que contiene los controladores REST que atenderán a las peticiones del Front-End que, utilizando la capa modelo, construirán las respuestas a las mismas.

La comunicación se realiza mediante REST, un estilo arquitectónico que emplea el protocolo HTTP para la obtención de datos y es comunmente usado para aplicaciones distribuidas.

La arquitectura Rest define varios métodos de acceso que especifican el tipo de acción sobre un recurso. El método GET solicita una representación de un recurso pedido, POST crea una representación de un recurso y PUT lo modifica. DELETE elimina el recurso especificado.

La distribución en paquetes del lado servidor se ha realizado por capas. El paquete base es *es.udc.tfg*.

- *es.udc.tfg.model*: Capa modelo. Contiene las capas de dominio, de acceso a datos y de

lógica de negocio.

  - *es.udc.tfg.model.entities*: Capa de dominio.
  - *es.udc.tfg.model.repositories*: Capa de acceso a datos.
  - *es.udc.tfg.model.service*: Capa de lógica de negocio.

- *es.udc.tfg.web*: Capa web que contiene la API REST.

Además, hay otros paquetes como config, util o exceptions que contienen elementos que se comparten entre capas o que se usan frecuentemente pero no forma parte de ninguna capa en concreto.

5.2.2 Arquitectura de lado cliente

La arquitectura del lado cliente es una arquitectura basada en componentes. Recibe los datos del lado servidor y presenta la vista en función de los datos recibidos. La distribución por módulos va ligada al estado propio de React y su navegabilidad entre componentes.

Así podemos diferencias los siguientes paquetes dentro del lado cliente.

- *backend*: Conexión con el lado servidor a través de la API REST
- *i8n*: Ficheros de internacionalización
- *modules*: Paquetes donde se encuentran los componentes de cada módulo.
- *store*: Inicialización del arbol de estado de la aplicación.

5.2.3 Internacionalización

La aplicación está diseñada de manera que pueda adaptarse a diferentes idiomas y regiones sin realizar cambios en el código.

Para lograrlo, se ha usado la librería *React-intl* que permite la internacionalización de aplicaciones React mediante ficheros JSON con el nombre de cada idioma. El idioma y la región son detectados por navegador.

Capítulo 6

Planificación

En este capítulo se expone la planificacion y el presupuesto inicial del proyecto.

6.1 Planificación inicial

Una vez conocidos los objetivos y herramientas de trabajo se realizará la planificación del proyecto. Para ello se hará un análisis de los recursos disponibles para el desarrollo.

Debido a la naturaleza del proyecto, existe una limitación de recursos, debido a que las tareas serán abordadas por una sola persona. Los equipos de desarrollo suelen estar formados por más de una persona y las tareas son distribuidas y realizadas en paralelo, disminuyendo la duración del proyecto.

Como la metodología a seguir en el proyecto es Scrum, el desarrollo de este se ha dividido en Sprints en los cuales se distribuyen las tareas en función de su prioridad.

Los siete Sprints realizados comprenden 120 puntos. Cada punto equivale aproximadamente a dos horas y media de trabajo.

$$120 * 2,5h = 300h \tag{6.1}$$

Por lo que el proyecto se realizará en siete Sprints con 300 horas de trabajo.

6.2 Presupuesto

Para evaluar costes de un proyecto es necesario tener en cuenta los gastos de personal, gastos de material y de licencias.

Debido a que el proyecto ha sido desarrollado completamente con tecnologías de software libre y gratuitas y que el proyecto es llevado a cabo en un ordenador personal no se tienen en cuenta costes sociales ni indirectos.

Siendo el precio del ordenador personal 800€ con un año de antiguedad, su tiempo de uso durante el desarrollo del proyecto 18 semanas y su vida útil cinco años (60 meses), la amortización del equipo es:

$$\frac{800€ - \frac{800€}{60meses} * 12meses}{60meses} * 4,5meses = 48€$$

Licencias: 0€

Servidor Web: 0€

Este proyecto cuenta con dos recursos humanos:

1. Una Jefa de proyectos: Patricia Martín Rodilla. Encargada de la toma de decisiones sobre la supervisión y evaluación del proyecto.
2. Un analista y programador: Carlos Fraiz. Encargado del diseño, desarrollo y realización de pruebas en el proyecto.

Según el informe emitido por Hays en 2020 [43] el coste de dichos perfiles es en torno a 25.000€ brutos anuales en el caso del Analista programador (perfil *Java developer*) y alrededor de 36.000€ anuales brutos en el de Jefa de proyecto (perfil *Team lead developr*). Este estudio sirvió de base para la estimación de costes de personal en el proyecto.

Recursos	**Coste**	**Tiempo de trabajo**	**Total**
Jefa de Proyecto	18€/h	24h	432€
Analista programador	12,5€/h	300h	3750€
Equipo personal	800€	4 meses y medio	48€
Licencias y servidor web	0€	-	0€
-			4500€

6.3 Seguimiento

Como se comentó anteriormente, el proyecto se realizará en Sprints. Se definieron un total de siete Sprints con duración variable. Los detalles de la ejecución concreta de los Sprints se explicarán en la sección 7: implementación. A continuación se describirán brevemente los Sprints a realizar.

6.3.1 Sprint 1

El objetivo de este Sprint fue el estudio del dominio y el diseño inicial de la aplicación. Debido a que no había requisitos en cuanto a las tecnologías a usar en el proyecto, también se estudiaron las diferentes alternativas que mejor se adaptaran al desarrollo. Se realizó el modelo de datos y se configuró el entorno.

6.3.2 Sprint 2

En este Sprint se completó la historia *Análisis de necesidades de visualización de datos* y se complementó con la formación en la librería D3.js, la cual iba a ser el eje central del desarrollo del proyecto.

6.3.3 Sprint 3

Este Sprint se completaron las historias de usuario *como usuario de la aplicación quiero poder crear, gestionar y usar mi cuenta para hacer uso de las funciones de la aplicación* y *como usuario identificado quiero poder añadir nuevos conjuntos de cuentas para poder clasificarlas y visualizar la información.*

6.3.4 Sprint 4

El cuarto Sprint se dedicó a la implementación de la historia de usuario *Como usuario identificado quiero ver una representación de las cuentas de twitter para diferenciar entre las clasificadas como bots y las clasificadas como humanos.*

6.3.5 Sprint 5

El quinto Sprint se centró en las historias de usuario de *Como usuario identificado quiero filtrar las cuentas introducidas según su porcentaje de bot o humano para poder buscar las cuentas con características especificas.*

6.3.6 Sprint 6

Este Sprint se dedicó a la historia de usuario *Como usuario identificado quiero poder ver grafos de relaciones entre las cuentas para poder detectar redes de bots.*

6.3.7 Sprint 7

En este último Sprint se finalizó la redacción de la memoria.

En todos los Sprints se realizaron las pruebas correspondientes de los casos de uso implementados y se redactó las partes de la memoria correspondientes.

Capítulo 7

Implementación

En este capítulo se detallará todo el trabajo realizado durante el desarrollo de la aplicación web. Se comentarán los aspectos más reseñables de la implementación de cada una de las historias de usuario de cada Sprint en orden cronológico. Únicamente se explicará código específico de las aportaciones algorítmicas originales del proyecto y se citarán los repositorios o páginas que hemos usado como referencia.

7.1 Sprint 1

El primer sprint se centró en realizar un diseño inicial de la aplicación: Modelo de datos, elección de arquitectura, definir necesidades de visualización, etc. También se realizó la elección de tecnologías para el *Back-end* y la configuración del entorno de desarrollo y las herramientas. Tuvo lugar entre los días 10 y 25 de Febrero.

7.2 Sprint 2

El segundo Sprint tuvo lugar entre el 25 de Febrero y el 18 de Marzo y se realizaron las historias de usuario de *análisis de necesidades de visualización* y *formación en el uso de la librería d3.js*. La formación en la librería consistió en probar las distintas funcionalidades de la librería. Algunos de los tutoriales y páginas para el aprendizaje que se siguieron son [44], [45].

7.3 Sprint 3

Debido a motivos externos el trabajo de fin de grado fue paralizado durante tres meses. El tercer Sprint transcurre por tanto entre el 7 y el 25 de Junio. En este Sprint se desarrollaron las historias de usuario *Como usuario de la aplicación quiero poder crear, gestionar y usar mi cuenta para hacer uso de las funciones de la aplicación* y *Como usuario identificado quiero poder añadir nuevos conjuntos de cuentas para poder clasificarlas y visualizar la información.*

En este Sprint se desarrollo la integración del clasificador en el *Back-end* de nuestra aplicación.

El clasificador es un conjunto de scripts escritos en lenguaje *perl* el cual es llamado desde consola de comandos pasándole un archivo en formato *xml* y devuelve como salida los nombres de las cuentas con los resultados del clasificador.

Para poder hacer uso del clasificador, ha sido necesario buscar la manera más eficaz y eficiente de llamar al clasificador. Para ello, se ha usado la clase ProcessBuilder de Java para crear un proceso en el sistema operativo y ejecutar el Script.

Este método genera una dependencia con el sistema operativo subyacente, ya que la ejecución de comandos es dependiente de cada *SO*.

```
1 ProcessBuilder builder = new ProcessBuilder("/bin/sh", "-c",
2     "cat input/" + filePath + " |/usr/bin/perl scripts/classif.perl
      > output/result.txt")
3
4 // Ponemos como directorio desde el que ejecutar el comando la
      carpeta donde
5 // tenemos los inputs, el script y las salidas
6 builder.directory(new File("classifier"));
```

Una vez finalizado el proceso, se procesan todos los datos de entrada y se persisten. Para ello se ha usado la librería *javax.xml.parsers* para ir cogiendo los diferentes datos relativos a las cuentas y tweets. Una vez persistido el fichero de entrada se persiste el fichero de salida el cual contiene los resultados del clasificador y al finalizar, crea el conjunto con toda las cuentas clasificadas.

El servicio dedicado a almacenar el fichero xml para poder ser clasificado tiene una configuración a través de un fichero *.properties* que permite configurar parámetros como tamaño máximo de archivos (los archivos con todas las cuentas y *Tweets* alcanzan tamaños de gigaby-

tes), tamaño mínimo, etc. Una vez clasificado el archivo es borrado.

En cuanto a la persistencia, destacar que se ha decidido no añadir en el *mappeo* objeto-relacional la relación de Cuentas de Twitter con los *Tweets*, con los me gusta y con los *Retweets* debido a la posible carga que resultaría de traer todos esos objetos con cada una de las cuentas.

7.4 Sprint 4

En el cuarto Sprint se desarrolló la historia de usuario *Como usuario identificado quiero ver una representación de las cuentas de twitter para diferenciar entre las clasificadas como bots y las clasificadas como humanas* y transcurre entre el 26 de junio y el 12 de julio.

En este sprint se desarrolló la visualización de *hexagonal grid & heat map*. Para ello partimos del código de [17]. De este artículo se usó el cálculo para obtener los puntos centrales del hexágono. Se calcula el ”radio” del hexágono del que se obtiene el círculo que toca sus seis esquinas a partir del número de hexágonos que queramos distribuir en columnas y filas. Como las filas impares tienen un desfase frente a las filas pares, es necesario hacer distinción en el eje horizontal dicho desfase.

También se ha modificado respecto al algoritmo original el orden en el que se generan los hexágonos en el componente. Originalmente se generaban de fila en fila, lo que implicaba que las cuentas, que recibíamos del *Back-end* ordenadas de bot a humano, se distribuían las bot en la parte superior del *grid* y las humanas en la parte inferior. En el capítulo 5: Diseño habíamos escogido distribuir las cuentas bot hacia la izquierda y las humanas hacia la derecha pues era una distribución más visual, y por ello se ha cambiado el algoritmo.

```
1 for (var i = 0; i < MapColumns; i++) {
2         for (var j = 0; j < MapRows; j++) {
3           var x = hexRadius * i * Math.sqrt(3)
4           if (j % 2 === 1) x += (hexRadius * Math.sqrt(3)) / 2
5           var y = hexRadius * j * 1.5;
```

Para elegir la cantidad de cuentas por fila y/o columna se contemplaron varias opciones: el mismo número de filas que de columnas (matriz cuadrada), un número fijo de cuentas por fila y la opción escogida fue obtener una matriz con mayor número de columnas que de filas y para la distribución exacta se usó el siguiente cálculo.

```
var MapColumns = Math.floor(Math.sqrt(cuentas.length)) +
  Math.floor(Math.sqrt(Math.floor(Math.sqrt(cuentas.length)))),
      MapRows = Math.floor(cuentas.length / MapColumns) +
  Math.ceil((cuentas.length - MapColumns *
  Math.floor(cuentas.length / MapColumns)) / MapColumns);
```

Esta distribución garantiza la escalabilidad en función del número de cuentas y facilitar la visualización general de los resultados del clasificador.

Puesto que la generación de hexágonos se hace de manera algorítmica se ignora el hecho de que una de las filas va a estar incompleta en hexágonos. Para esto se añadió al algoritmo de generación de hexágonos las siguientes condiciones cuando estuviese generando los de la última fila.

```
if (j + 1 == MapRows && MapColumns * MapRows > cuentas.length) {
            if (j % 2 == 1 && cuentas.length % MapColumns > i) {
              points.push([x, y]);
            }
            else if (j % 2 == 1)
              continue
            else if (i < cuentas.length % MapColumns)
              points.push([x, y]);
            else
              continue;
          } else
            points.push([x, y]);
```

La distribución de hexágonos para nuestra casuística del clasificador partiendo de una implementación previa del *hexagonal grid and heat map* es una contribución original y se pondrá a disposición de la comunidad en repositorios de open source.

7.5 Sprint 5

En el quinto Sprint se implementaron las historias de usuario *Como usuario identificado quiero ver una representación de las cuentas de twitter para diferenciar entre las clasificadas como bots y las clasificadas como humanos* y transcurre entre el 13 y el 30 de julio.

Con respecto a la primera historia, se estudiaron dos opciones. Realizar una llamada al *Back-end* para traer las cuentas que cumplen con las condiciones especificadas en el filtro (solo bots, solo humanas, todas y con un porcentaje mínimo) o filtrar las cuentas en el Front-end y

almacenar el conjunto de todas ellas por separado de las cuentas que usa la visualización.

La segunda alternativa genera un menor número de peticiones al *Back-end* pero aumenta en gran medida la complejidad de la lógica en el lado cliente, lo que puede relentizar la visualización. Se optó por la primera opción dando prioridad a mantener la lógico e inviolabilidad de las capas definidas en la fase de diseño.

En el *Back-end* cabe destacar Íla implementación de un método en el *DAO*. Todos los métodos usados hasta este momento eran heredados de los *DAOs* genéricos que proporciona Spring, pero como la consulta en este caso podía contener o no algunos elementos (filtro solo por porcentaje o solo por tipo de cuenta), era un caso más complejo. El método recupera las cuentas de Twitter que cumplen los parametros especificados de porcentajes del conjunto cuyas cuentas estamos visualizando en la vista.

```
1
2  String queryString = "SELECT distinct c FROM CuentaTwitter c join
     c.resultadosClasificador r WHERE r.conjunto.id  = :idConjunto";
3
4    if (porcentajeBot != null) {
5      queryString += " AND ((r.porcentajeBot IS NOT NULL AND
     r.porcentajeBot >= :porcentajeBot)";
6    }
7
8    if (porcentajeBot != null && porcentajeHumano != null) {
9      queryString += " OR";
10   } else if (porcentajeHumano != null) {
11     queryString += " AND";
12   }
13
14   if (porcentajeHumano != null) {
15     queryString += " (r.porcentajeHumano IS NOT NULL AND
     r.porcentajeHumano >= :porcentajeHumano)";
16   }
17
18   if (porcentajeBot != null)
19     queryString += ")";
```

7.6 Sprint 6

El sexto Sprint se dedicó a la implementación de la historia de usuario *Como usuario identificado quiero poder ver grafos de relaciones entre las cuentas para poder detectar redes de bots*

y transcurre entre el 31 de julio y el 17 de agosto.

Como en el caso de la primera visualización, la implementación fue acompañada del diseño, aplicando patrones de InfoVis (Por ejemplo *focus+context*) y estudiando las necesidades de los usuarios finales.

En relación al *Back-end*, durante el desarrollo se decidió recuperar de la capa de acceso a datos el número de interacciones entre cada una de las cuentas del conjunto con las demás. Para ello, se creó un método que contabilizaba estas interacciones, pero se observó que la carga de todas las peticiones necesarias generaba unos tiempos de espera en la generación de la visualización que no eran aceptables.

Para solucionar este problema, se creo una nueva entidad **interacción**. Esta entidad consta de los nombres del autor de la interacción y del receptor, el tipo de interacción y el número de interacciones de este tipo. De esta manera, una sola petición a la base de datos podía devolver una lista de interacciones, eliminando los tiempos de espera derivados de las llamadas de Hibernate a la base de datos.

```
1
2  public List<Interaccion> retweetsEntreCuentas(List<String> cuentas)
       {
3      String queryString = "SELECT NEW
       es.udc.tfg.backend.model.entities.Interaccion(
4      t.usuario.nombreUsuario, r.nombreUsuario, count(t), 'Retweet')"
5          + " FROM Tweet t join t.retweet r WHERE
       t.usuario.nombreUsuario in :cuentas"
6          + " AND r.nombreUsuario in :cuentas AND r.nombreUsuario not
       like t.usuario.nombreUsuario"
7          + " group by t.usuario.nombreUsuario, r.nombreUsuario";
8
9      Query query = entityManager.createQuery(queryString);
10     query.setParameter("cuentas", cuentas);
11     List<Interaccion> interacciones = query.getResultList();
12     return interacciones;
```

En la capa *Front-end* se basó el desarrollo en estos dos proyectos principalmente [19] y [46].

La principal aportación fue la interacción con los nodos y enlaces.

```
1
2  .on("dblclick", function (c) {
3          const nodes = new Set();
4          const links = new Set();
```

```
5           d3.selectAll(".links")
6             .attr("stroke", function (l) {
7               if (l.source.id != c.id && l.target.id != c.id) {
8                 // eslint-disable-next-line no-restricted-globals
9                 d3.select(this).attr("marker-end", `url(${new
    URL(`#arrow-Undefined`, location)})`);
10                return "#DCDCDC80";
11              }
12              else {
13                nodes.add(l.source.id)
14                nodes.add(l.target.id)
15                links.add(l)
16                // eslint-disable-next-line no-restricted-globals
17                d3.select(this).attr("marker-end", `url(${new
    URL(`#arrow-${l.interacción}`, location)})`);
18                return color(l.interacción)
19              }
20            })
21          d3.selectAll(".ndc")
22            .attr("fill", function (d) {
23              if (nodes.has(d.id))
24                return colorScale(d.resultado)
25              else
26                return "#DCDCDC80"
27            });
28
29          simulation.stop()
30
31
32
33          d3.select(".c-username")
34            .classed("invisible", false)
35            .select(".userName")
36            .html("@" + c.id);
37          d3.select(".c-username")
38            .select(".porcentaje")
39            .html(c.resultado * 100 + "%");
40
41
42          text.text(function (d, i) {
43            if (d.source.id == c.id || d.target.id == c.id)
44              return d.quantity
45          }).attr("fill", d => color(d.interacción));
46
47        });
```

La distribución de grafo para nuestra casuística del clasificador partiendo de una implementación previa del *force-directed graph* es una contribución original y se pondrá a disposición de la comunidad en repositorios de open source.

7.7 Sprint 7

Este sprint se dedicó a la realización de pruebas, evaluiación informal con usuarios y documentación final, entre la que se encuentra la redacción de la presente memoria. Se realizó entre el 18 de agosto y el 3 de septiembre.

7.8 Control de versiones

Para el desarrollo del proyecto y la gestión de los cambios se ha usado Git. Gracias a su uso disponemos de un historial completo de los cambios realizados en la aplicación a lo largo de los Sprints y de cada una de las versiones liberadas en cada iteracción.

Capítulo 8

Pruebas

En este capítulo se comentarán las distintas pruebas realizadas durante el proyecto.

Las pruebas son una parte fundamental en el desarrollo software. Las pruebas están pensadas para detectar errores, no para demostrar la ausencia de ellos. No es posible alcanzar un software sin errores pero si aumentar nuestra confianza en su calidad.

8.1 Pruebas

Se realizaron pruebas de unidad, integración y aceptación sobre la aplicación desarrollada.

8.1.1 Pruebas de unidad

La integración del clasificador en el Back-end era un punto clave en el desarrollo del proyecto debido a que se trata de código heredado en un lenguaje diferente y de un entorno de investigación en lugar de uno de producción empresarial. Por ello, para una mayor confianza en la implementación del servicio que llamaba al proceso, se realizaron tests de unidad con la librería Mockito para poder probar su correcto funcionamiento en distintos escenarios.

Los test prueban la llamada al clasificador usando instancias falsas (*mocks*) de los demás componentes (Servicios y DAOs) permitiendo así pre-definir una salida a las llamadas de dichas clases.

Debido a que el clasificador necesita ficheros de cuentas en formato *xml*, se han usado

ficheros preparados para las pruebas.

8.1.2 Pruebas de integración

Para las pruebas de integración entre los distintos componentes del sistema se usó el framework de pruebas JUnit tal y como se comentó en el apartado de fundamentos tecnológicos y herramientas utilizadas 4. JUnit está formada por un conjunto de librerías que permiten la creación y ejecución de pruebas comprobando el correcto funcionamiento de los métodos.

Se han hecho pruebas de integración de los servicios (capa lógica de negocio) a medida que se desarrollaba el proyecto. Los test se encuentran en el directorio *src/test/java*.

Para el contexto de pruebas se ha utilizado una base de datos distinta a la usada para producción.

8.1.3 Pruebas de calidad interna

De manera complementaria se realizaron pruebas para detectar prácticas no recomendadas que pueden llevar a *bugs* con la herramienta *SpotBugs*. Se ha usado como extensión de Eclipse (Figura 8.1).

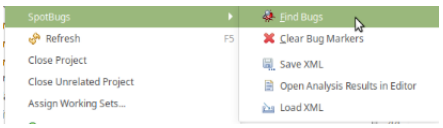


Figura 8.1: Menú de la extensión *SpotBugs* en eclipse

Como se muestra en la figura 8.2, la herramienta muestra prácticas que pueden llevar a errores en la aplicación, como por ejemplo la comparación de dos valores Long mediante el operador '==', siendo más recomendable el uso del método 'Equals'.

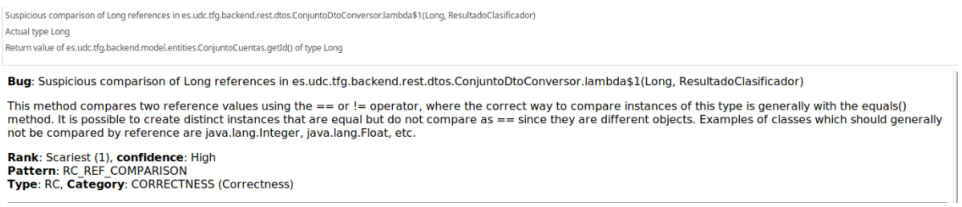


Figura 8.2: Aviso de *SpotBugs* por una mala práctica

8.1.4 Pruebas de escalabiliadad

Se han realizado pruebas con diversos volúmenes de cuentas y configuraciones de las visualizaciones para comprobar como se adaptan nuestros sistemas de InfoVis. En el futuro son necesarias más pruebas automatizadas de escalabilidad y volumen de datos en las visualizaciones.

Capítulo 9

Producto final

En este capítulo se muestra las funcionalidades de la aplicación implementada y se detalla su funcionamiento. Esta sección servirá como guía de uso y describirá las características principales del software desarrollado. La sección irá acompañada de capturas que servirán de apoyo visual para la explicación.

9.1 Gestión de usuario

La gestión de usuarios es intuitiva y básica. Permite registrarse antes de acceder a cualquier recurso (Figura 9.1) o iniciar sesión una vez registrado con éxito (Figura 9.2).

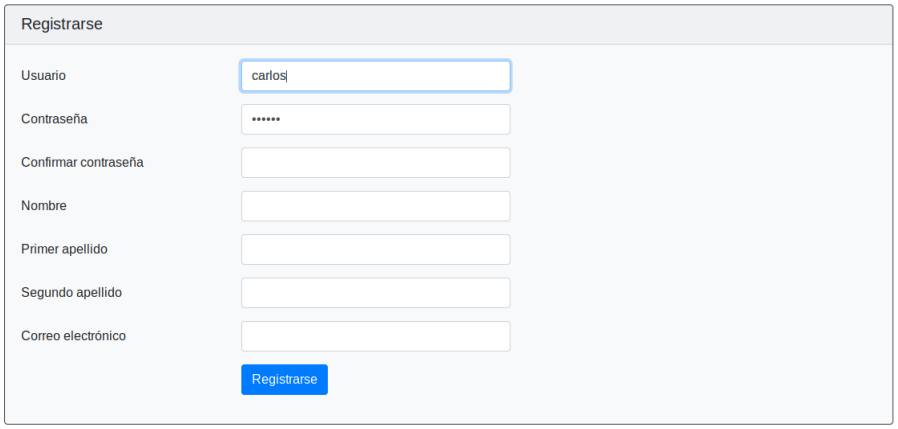


Figura 9.1: Formulario de registro

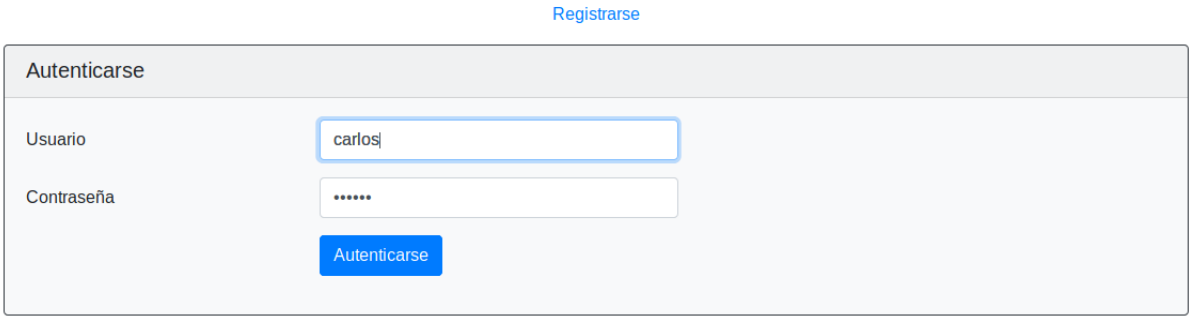


Figura 9.2: Formulario de inicio de sesión

También se dispone de la posibilidad de cambiar parámetros del perfil, como la contraseña en un formulario único (Figura 9.3), o el nombre correo o apellidos (Figura 9.4).

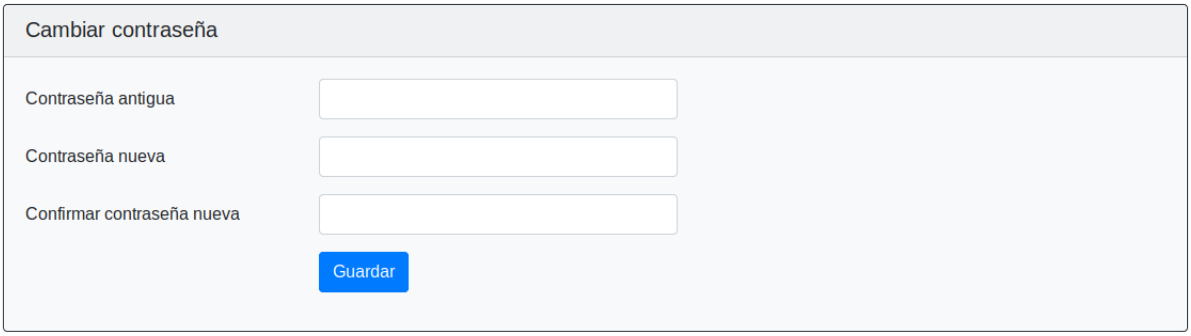


Figura 9.3: Formulario para cambio de contraseña

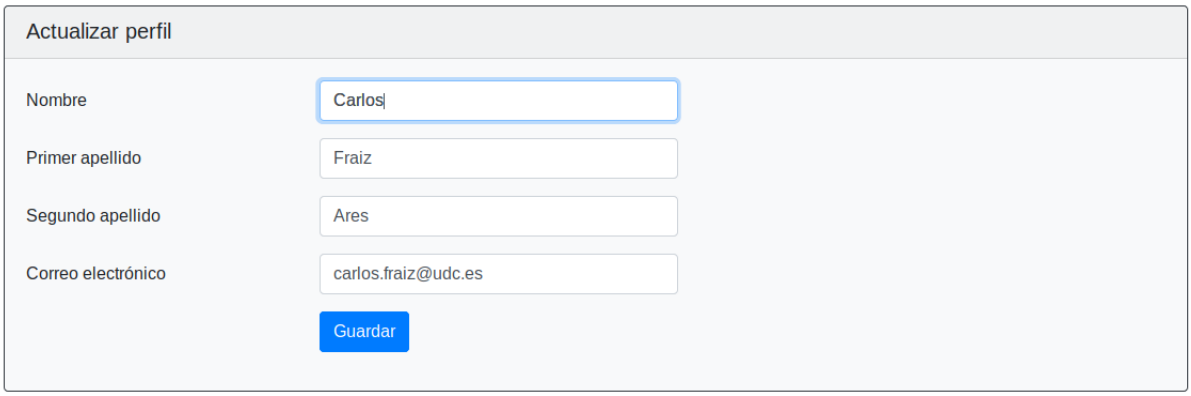


Figura 9.4: Formulario para cambio de datos del perfil

9.2 Clasificador

El usuario podrá acceder desde la pantalla principal al formulario para clasificar un fichero de cuentas. Debe ir acompañado de un nombre y opcionalmente una descripción (Figura 9.5).

Figura 9.5: Formulario para subir y clasificar el fichero de cuentas

Una vez importado y clasificado, se muestran los detalles del conjunto creado (Figura 9.6).

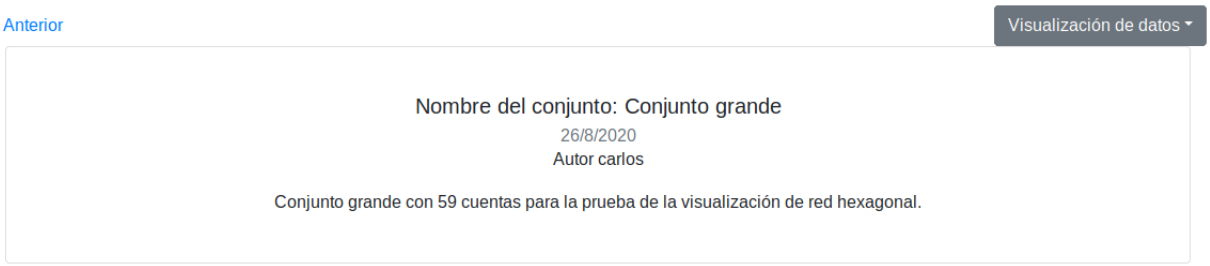


Figura 9.6: Detalle del conjunto seleccionado

Adicionalmente, puedes acceder a todos los conjuntos creados ordenador por fecha de creación para visualizar distintos conjuntos de datos (Figura 9.7).

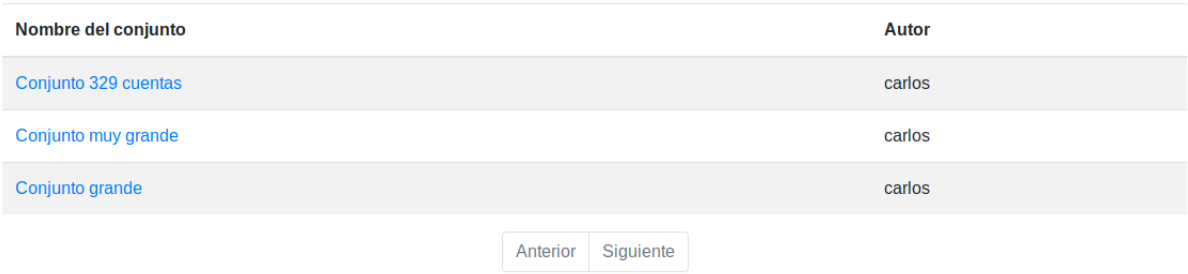


Figura 9.7: Lista de conjuntos creados ordenados por fecha de creación

9.3 Visualizaciones

Una vez en los detalles del conjunto, podemos acceder a las dos visualizaciones principales (Figura 9.8).

Figura 9.8: Acceso a los sistemas de InfoVis desde los detalles del conjunto

9.3.1 Visualización de hexágonos

La visualización de hexágonos permite visualizar todas las cuentas usando los colores para indicar con que certeza una cuenta es un bot o una cuenta humana. El tamaño de los hexágonos se calcula de manera dinámica, de manera que los grandes conjuntos de cuentas se puedan diferenciar y analizar, pero permitiendo hacer microanálisis de conjuntos más pequeños (Figura 9.9).

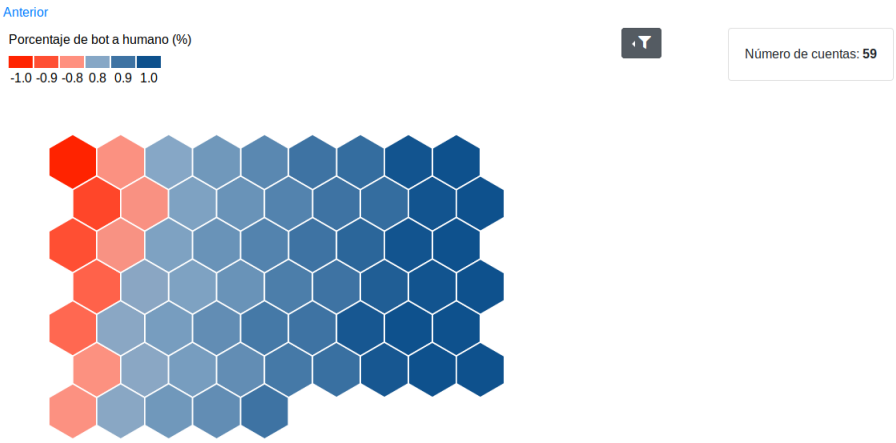


Figura 9.9: Visualización de *hexagonal grid and heatmap*

Al interaccionar con los hexágonos aparece una ventana dando detalles disponibles de las cuentas (Figura 9.10).

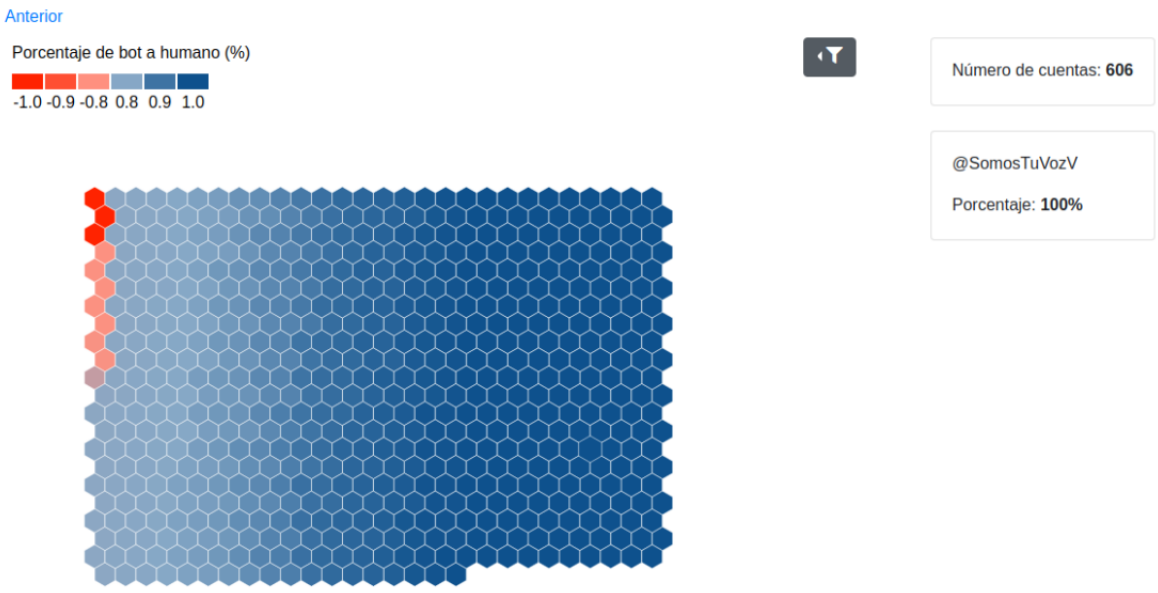


Figura 9.10: Interacción con los hexágonos

El filtro (Figura 9.11) permite filtrar todas las cuentas para solo mostrar aquellas que cumplan los criterios seleccionados (Figura 9.12).

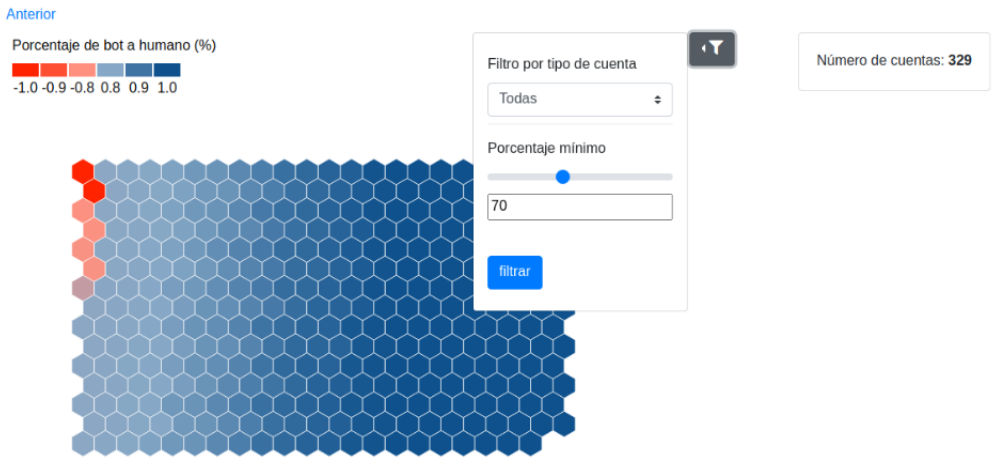


Figura 9.11: Menú del filtro de cuentas

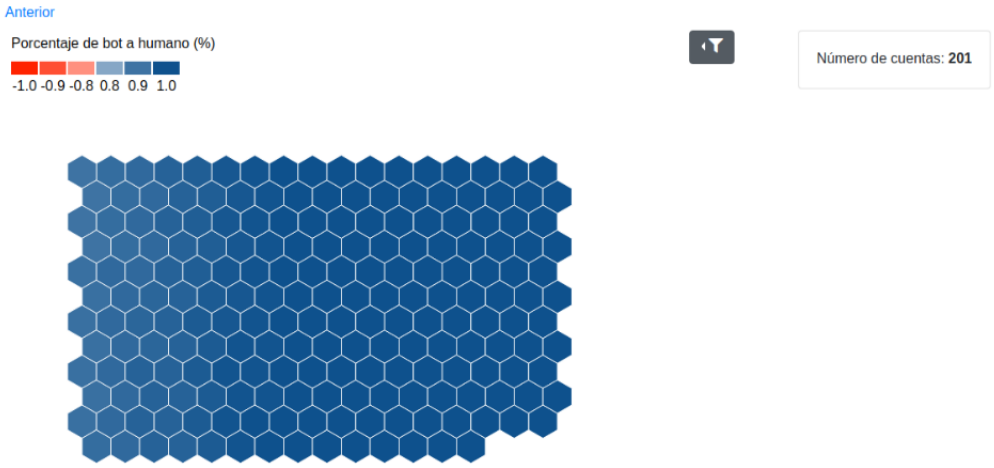


Figura 9.12: Cuentas solo humanas y con más de 70% de probabilidad

9.3.2 Visualización de grafo

Una vez cargada la visualización de grafo podemos observar todas las cuentas representadas en nodos de tamaño variable en función de cuantas interacciones reciben del resto de cuentas del conjunto (Figura 9.13).

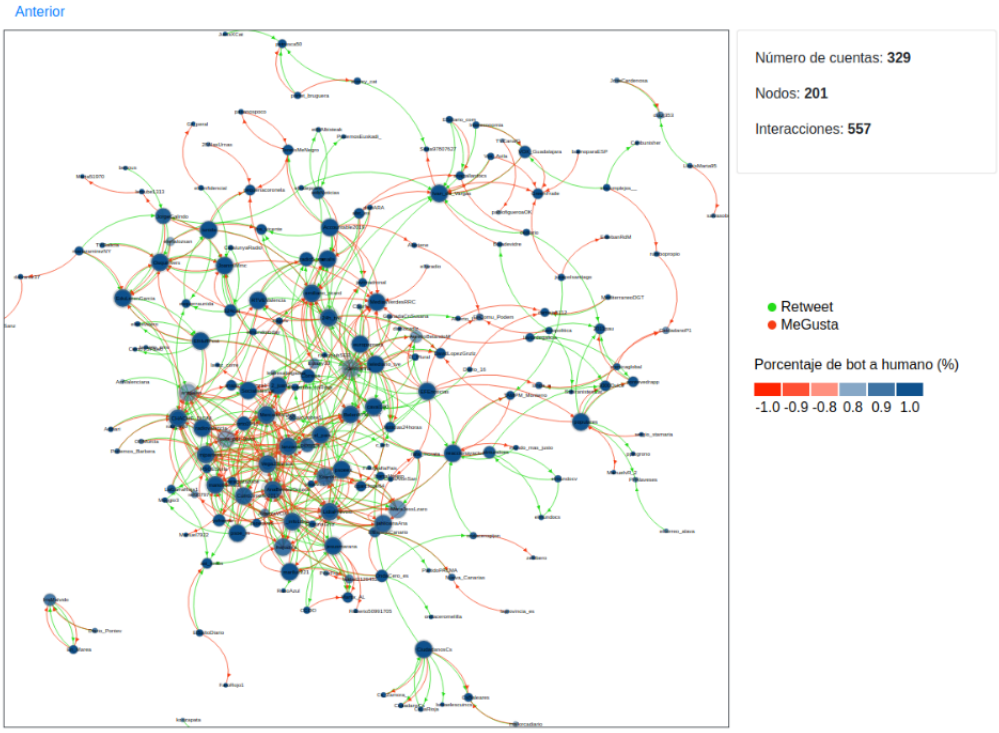


Figura 9.13: Visualización de *force-directed graph*

Los enlaces representan las interacciones entre las cuentas y su color identifica el tipo de relación.

Al interaccionar con un nodo, todos los demás nodos se ponen en segundo plano menos con el que interaccionamos y todos aquellos con los que haya interaccionado (Figura 9.14). Debido a la gran cantidad de información que se muestra, la visualización cuenta con un zoom que permite navegar entre los nodos y enlaces (Figura 9.15).

Al igual que en la visualización de red hexagonal, se muestran detalles de la cuenta al interaccionar con un nodo.

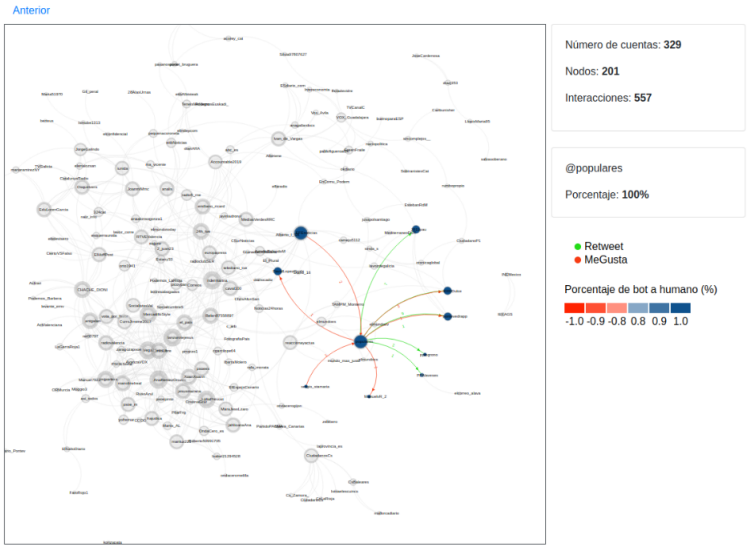


Figura 9.14: Interacción con los nodos

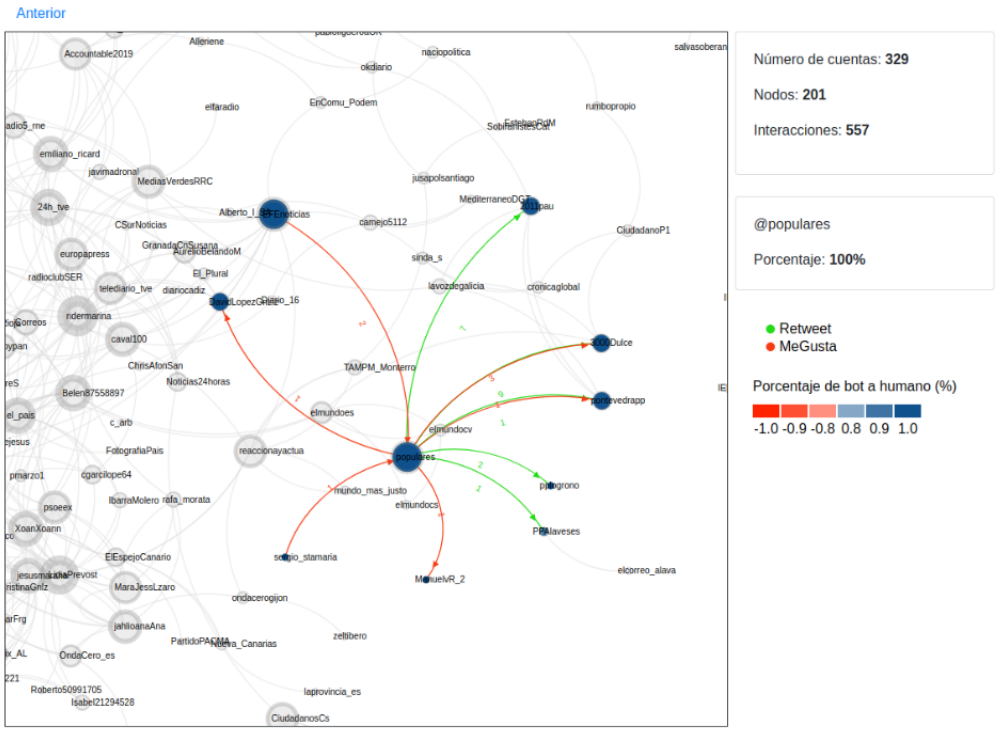


Figura 9.15: Zoom en la visualización de *force-directed graph*

Capítulo 10

Conclusiones

En este capítulo final se hará una evaluación crítica sobre el desarrollo del proyecto y se detallarán líneas de mejora futuras que aportarían mayor valor al producto.

10.1 Desarrollo del proyecto

Una vez terminado el proceso de desarrollo se cumplieron los objetivos propuestos y especificados en el capítulo 6: Planificación. Al final de este desarrollo hemos obtenido una aplicación web que adapta el clasificador original a entornos webs modernos e implementa técnicas de InfoVis para proporcionar a los expertos un sistema en el cual analizar los datos siguiendo sus preguntas de investigación. Específicamente:

- Permite la creación y gestión perfiles de usuario siguiendo tres roles para acceder a las funcionalidades de la aplicación.
- Permite hacer uso del clasificador de manera intuitiva y sencilla persistiendo los datos y añadiendo información a los conjuntos de cuentas.
- Proporciona un listado e historial de todos los conjuntos de la aplicación.
- Permite acceder a la información en detalle de cada uno de los conjuntos que permite cierto trabajo colaborativo, pudiendo visualizar conjuntos analizados de otros investigadores.
- Proporciona una visualización de *hexagonal grid* que permite analizar los resultados del clasificador en el conjunto de cuentas.

- Proporciona una visualización *force-directed graph* que responde a las preguntas sobre como las cuentas en el conjunto interactuan entre ellas.

10.2 Lecciones aprendidas

Durante el desarrollo del proyecto se adquirieron conocimientos de gran importancia en las tecnologías utilizadas (Ver capítulo 4), especialmente en D3.js y en el campo de la *InfoVis*. Adicionalmente, se profundizó en tecnologías ya usadas anteriormente como React, redux, *Spring* (en especial *Spring security* o Java.

La *InfoVis* es un sector cada vez más amplio con ramificaciones en tecnología web, pero también tecnología móvil, entornos inmersivos, etc., y de gran importancia en perfiles laborales como *data scientist*, *data engineer*, *UX developer*, etc. Por lo que los conocimientos adquiridos tienen un gran valor en el mercado laboral.

En cuanto a la planificación, se pusieron en práctica los conocimientos adquiridos durante la carrera en un proyecto real. Cada vez más empresas y proyectos aplican metodologías ágiles en sus proyectos, y ponerla en práctica ha dado un conocimiento más en profundidad sobre ella.

10.3 Líneas futuras

En el desarrollo se completaron todos los objetivos propuestos en la planificación, pero teniendo en cuenta la naturaleza del proyecto (responder a las preguntas de investigación realizadas por los usuarios) se podrían añadir nuevas funcionalidades o ampliar las ya existentes para enriquecer las respuestas proporcionadas. Algunos ejemplos son:

- Como valor añadido para los usuarios sería interesante ver distintos detalles de las cuentas y los conjuntos para poder hacer un mejor *micro-análisis*. Algunos ejemplos serían:
  - Visualización complementaria dentro de la aplicación que permite ver los cuerpos de texto de los tweets de cada cuenta para poder ver el tipo de contenido o mensajes que publican.
  - Estadísticas de los conjuntos tales como *Hashtags* más usados, *Tweets* más *retweeteado* o con más me gusta, etc.

  - La integración con la API de twitter permitiría recoger información sobre las cuentas de la que no dispongamos mediante el fichero introducido por los usuarios. Información tal como la biografía, sus seguidores, su foto de perfil, etc. Podría ser de interés para hacer análisis individual de cuentas concretas.

- En el caso de la visualizaciones, durante la fase de diseño (Ver capítulo 5) se valoraron otras alternativas que se descartaron al priorizar las dos implementadas. Algunos ejemplos serían.

  - ***Geo-visualization***: La información de como los bots políticos inciden en ciertas regiones en el tiempo podría aportar información útil a los expertos sobre los intereses o el objetivo de las distintas redes de bots usadas por los agentes políticos.
  - ***Cloud-Visualization***: Los hashtags forman parte fundamental de la capacidad de difusión de Twitter, y por ello forman un elemento fundamental en el análisis político de las cuentas de *Twitter* y las redes de bots. La visualización consistiría en un conjunto de Hashtags distribuidos usando el elemento del tamaño para indicar el número de menciones a este y el color que tipo de cuentas (bots o humanas) han usado más el hashtag [47].

- Inclusión de transiciones naturales en las visualizaciones para adaptar mejor los cambios en volumen de datos o en fuente [48].

- Realizar estudios empíricos reglados por parámetros de ingeniería del software para determinar grado de eficiencia, productividad y satisfacción de los usuarios con la aplicación [49]. Esto es especialmente relevante en usuarios de perfil investigador y con perfiles no tecnológicos [50].

- Incluir más idiomas en la internacionalización, ya que la aplicación únicamente dispone de castellano e inglés habilitados.

Bibliografía

[1] F. Windhager and M. Smuc, "The arts of the possible," 2014. [Online]. Available: http://www.jedem.org/index.php/jedem/article/view/308/283

[2] Szep and A. Jeno, "Algorithmic analysis visualization of bots' influence on us elections your vote," 2014. [Online]. Available: https://repository.arizona.edu/bitstream/handle/10150/632790/azu_etd_hr_2019_0227_sip1_m.pdf

[3] M. L. Congosto, "Elecciones europeas 2014: Viralidad de los mensajes en twitter," 2015. [En línea]. Disponible en: https://www.redalyc.org/pdf/931/93138738002.pdf

[4] C. Wardle, "Fake news. it's complicated." 2018. [En línea]. Disponible en: https://firstdraftnews.org/latest/fake-news-complicated/

[5] K. B. M. L. Arkaitz Zubiaga, Ahmet Aker and R. Procter, "Detection and resolution of rumours in social media: A survey," 2018. [En línea]. Disponible en: https://dl.acm.org/doi/pdf/10.1145/3161603

[6] A. Bessi and E. Ferrara., "Social bots distort the 2016 us presidential election online discussion." 2016. [En línea]. Disponible en: https://firstmonday.org/article/view/7090/5653

[7] S. Bradshaw and P. N. Howard, "Challenging truth and trust: A global inventory of organized social media manipulation," 2018. [En línea]. Disponible en: https://comprop.oii.ox.ac.uk/wp-content/uploads/sites/93/2018/07/ct2018.pdf

[8] A. M.-H. Michelle Forelle, Phil Howard and S. Savage, "Political bots and the manipulation of public opinion in venezuela." [En línea]. Disponible en: https://arxiv.org/ftp/arxiv/papers/1507/1507.07109.pdf

[9] J. M. Stuart K. Card and B. Shneiderman, *Readings in Information Visualization – Using Vision to Think*, 1st ed. Morgan Kaufmann, 1999.

[10] M. F. M. Luz Congosto and E. M. Egido, "Twitter y política: Información, opinión y ¿predicción?" 2011. [En línea]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/21631/twitter_congosto_EVOCA_2011.pdf

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[12] M. S. J. M. L. Dominik Dellermann M.Sc., Philipp Ebel, "Hybrid intelligence." [En línea]. Disponible en: https://link.springer.com/article/10.1007/s12599-019-00595-2

[13] E. Kamar, "Directions in hybrid intelligence: Complementing ai systems with human intelligence." [En línea]. Disponible en: https://hi4nlp.pages.citius.usc.es/

[14] Ecai 2020. [En línea]. Disponible en: https://hi4nlp.pages.citius.usc.es/#cfp

[15] Workshop on hybrid intelligence for natural language processing tasks (co-located at ecai-2020). [En línea]. Disponible en: https://digital.ecai2020.eu/

[16] Diansheng Guo, Jin Chen, A. M. MacEachren, and Ke Liao, "A visualization system for space-time and multivariate patterns (vis-stamp)," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1463–1464, 2006.

[17] N. Bremer. Creating hexagonal heatmaps with d3.js. [En línea]. Disponible en: https://www.visualcinnamon.com/2013/07/self-organizing-maps-creating-hexagonal.html

[18] R. Bourqui, D. Auber, and P. Mary, "How to draw clusteredweighted graphs using a multilevel force-directed graph drawing algorithm," 2007.

[19] M. Bostock. Mobile patent suits. [En línea]. Disponible en: https://observablehq.com/@d3/mobile-patent-suits

[20] "Mysql main page." [En línea]. Disponible en: https://www.mysql.com/

[21] "Hibernate main page." [En línea]. Disponible en: https://hibernate.org/

[22] "Spring overview and tutorial." [En línea]. Disponible en: https://www.tutorialspoint.com/spring/spring_overview.htm

[23] "Maven overview." [En línea]. Disponible en: https://maven.apache.org/

[24] "Junit overview." [En línea]. Disponible en: https://junit.org/junit5/docs/current/user-guide/

[25] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[26] S. Murray, *Interactive Data Visualization for the Web: An Introduction to Designing with D3*, 1st ed. O'Reilly Media, 2013.

[27] P. K. Janert, *D3 for the Impatient: Interactive Graphics for Programmers and Scientists*, 1st ed. O'Reilly Media, 2019.

[28] H. da Rocha, *Learn D3.js: Create interactive data-driven visualizations for the web with the D3.js library*, 1st ed. Packt Publishing, 2019.

[29] "React overview." [En línea]. Disponible en: https://reactjs.org/tutorial/tutorial.html

[30] "Redux overview and tutorial." [En línea]. Disponible en: https://redux.js.org/introduction/getting-started

[31] "Bootstrap overview and tutorial." [En línea]. Disponible en: https://getbootstrap.com/docs/4.5/getting-started/introduction/

[32] "Yarn overview and tutorial." [En línea]. Disponible en: https://yarnpkg.com/getting-started

[33] "Eclipse overview." [En línea]. Disponible en: https://www.eclipse.org/eclipse/eclipse-charter.php

[34] "Gitlab overview." [En línea]. Disponible en: https://about.gitlab.com/what-is-gitlab/

[35] "Taiga overview." [En línea]. Disponible en: https://project-management.com/taiga-software-review/

[36] "Visual studio code overview." [En línea]. Disponible en: https://code.visualstudio.com/docs

[37] "Latex overview." [En línea]. Disponible en: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes

[38] K. S. y Jeff Sutherland, *La Guía de Scrum.* Graphics Press, 2013. [En línea]. Disponible en: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

[39] J. P. Alexander Menzinsky, Gertrudis López, *Scrum Manager: Temario Troncal I.* Graphics Press, 2019. [En línea]. Disponible en: https://www.scrummanager.net/files/scrum_manager.pdf

[40] S. Bjork, "Redefining the focus and context of focus+context visualization," 2000. [En línea]. Disponible en: https://ieeexplore.ieee.org/document/885094

[41] A. S. Badashian, M. Mahdavi, A. Pourshirmohammadi, and M. M. nejad, "Fundamental usability guidelines for user interface design," in *2008 International Conference on Computational Sciences and Its Applications*, 2008, pp. 106–113.

[42] E. Tufte, *The visual display of quantitative information*, 2nd ed. Graphics Press, 2001.

[43] "Guia salarial trabajadores," 2020. [En línea]. Disponible en: https://cloud.email.hays.com/GuiaSalarial-trabajadores-2020

[44] J. Calzado. D3+react.js. [En línea]. Disponible en: https://lemoncode.net/lemoncode-blog/2018/7/16/react-y-d3js-trabajando-juntos-ii-soluciones

[45] S. Gutierrez. D3 tutorial from zero to binding data. [En línea]. Disponible en: https://www.dashingd3js.com/table-of-contents

[46] cliffng. Twitter users force-directed graph(network graph). [En línea]. Disponible en: https://observablehq.com/@cliffng/force-directed-graph-network-graph-with-arrowheads-and-lab

[47] AshwiniRS, "Wordcloud of tweets hashtags on blockchain topic." [En línea]. Disponible en: https://gist.github.com/AshwiniRS/c6b9cb999d80763061adef184677e291#file-wordcloud_hashtag-ipynb

[48] G. G. R. Jeffrey Heer, "Animated transitions in statistical data graphics." [En línea]. Disponible en: https://idl.cs.washington.edu/files/2007-AnimatedTransitions-InfoVis.pdf

[49] M. H. M. O. B. R. A. W. C. Wohlin, P. Runeson, *Experimentation in Software Engineering*. Springer-Verlag Berlin Heidelberg, 2012.

[50] C. G.-P. O. P. Patricia Martín-Rodilla, José Ignacio Panach, "Assessing data analysis performance in research contexts: an experiment on accuracy, efficiency, productivity and researchers' satisfaction," pp. 177–204, 2018. [En línea]. Disponible en: http://www.incipit.csic.es/en/research_outcomes/publication/artigo/assessing-data-analysis-performance-in-research-contexts-an-experiment-on-accuracy-efficiency-produc21